

Content Detection in Handwritten Documents

by

Shaik Mohammed Faizaan

A Thesis Presented in Partial Fulfillment  
of the Requirements for the Degree  
Master of Science

Approved June 2018 by the  
Graduate Supervisory Committee:

Kurt VanLehn, Co-Chair  
Salman Cheema, Co-Chair  
Yezhou Yang

ARIZONA STATE UNIVERSITY

August 2018

## ABSTRACT

Handwritten documents have gained popularity in various domains including education and business. A key task in analyzing a complex document is to distinguish between various content types such as text, math, graphics, tables and so on. For example, one such aspect could be a region on the document with a mathematical expression; in this case, the label would be math. This differentiation facilitates the performance of specific recognition tasks depending on the content type. We hypothesize that the recognition accuracy of the subsequent tasks such as textual, math, and shape recognition will increase, further leading to a better analysis of the document.

Content detection on handwritten documents assigns a particular class to a homogeneous portion of the document. To complete this task, a set of handwritten solutions was digitally collected from middle school students located in two different geographical regions in 2017 and 2018. This research discusses the methods to collect, pre-process and detect content type in the collected handwritten documents. A total of 4049 documents were extracted in the form of image, and json format; and were labelled using an object labelling software with tags being text, math, diagram, cross out, table, graph, tick mark, arrow, and doodle. The labelled images were fed to the Tensorflows object detection API to learn a neural network model. We show our results from two neural networks models, Faster Region-based Convolutional Neural Network (Faster R-CNN) and Single Shot detection model (SSD).

*A person with great patience is my brother, Abdul Mateen. I dedicate this thesis to my brother.*

## ACKNOWLEDGMENTS

This work would not have been possible without the scholastic and financial support from Dr. Salman Cheema and Dr. Kurt VanLehn. I am especially indebted to Dr. Salman, project lead of FACT, who has been supportive of this thesis and also my career goals. I am grateful to all those with whom I have had the pleasure to work during the time I worked for the development of the FACT classroom software.

I thank Khyati Raka for her immense interest in this thesis and her support in labelling data and writing the thesis document. I thank Taeyeong Choi for his help in lending a GPU laptop and for his conceptual support on training of the network. I would also like to thank Darshan Shetty and Usama Hashmi for their continuous moral support.

## TABLE OF CONTENTS

	Page
LIST OF TABLES .....	vi
LIST OF FIGURES .....	vii
CHAPTER	
1 INTRODUCTION .....	1
1.1 Terminology .....	4
1.2 Reader's Guide .....	4
2 Related Work .....	6
2.1 Preliminary Data Collection & Research .....	7
3 DATA COLLECTION .....	11
3.1 Methods .....	11
3.2 Tagging .....	13
3.3 Data Statistics .....	14
4 METHODOLOGY .....	17
4.1 Convolutional Neural Networks .....	17
4.2 Faster R-CNN .....	18
4.3 Single Shot Detector (SSD) .....	20
4.4 System Specifications .....	21
5 RESULTS .....	22
5.1 Intersection over Union (IoU) .....	22
5.2 Precision and Recall .....	23
5.3 ROC Curves .....	26
5.4 Mean Average Precision (mAP) .....	30
5.5 Time analysis .....	32
5.6 Example detections .....	33

CHAPTER	Page
6 Possible Improvements .....	38
6.1 Possible improvements in the model .....	38
6.2 Possible improvements in the data .....	38
6.3 Complication with the data.....	39
7 CONCLUSION .....	41
REFERENCES .....	42

## LIST OF TABLES

Table	Page
3.1 Labels given to a region on a handwritten notes . . . . .	15
5.1 The correlation between precision and recall with varying prediction score . . . . .	25
5.2 Time comparison of the two models on Nvidia Tesla K40m (12GB) . . . .	33

## LIST OF FIGURES

Figure	Page
1.1 A training handwritten sample of a solved math problem from FACT project .....	2
1.2 A detected handwritten sample of a solved math problem from FACT project .....	3
2.1 A labelled training sample of a solved math problem from FACT project	9
2.2 A test image of a solved math problem from FACT project detected by the neural network model.....	10
3.1 A sample solved problem from FACT project .....	12
3.2 A sample image of a poster containing only strokes .....	13
3.3 A sample image of a card containing only strokes.....	14
3.4 shows the number of labels of each class in our dataset .....	16
4.1 A typical supervised convolutional neural network architecture showing feature learning and classification phases.....	18
4.2 Architectural comparison of the two models. (a) Faster R-CNN has two networks - one to generate region proposals, the other to predict a class to find a box refinement for each proposal. (b) SSD has a single network to find regions, classify them and to refine to each region proposal. Images are from (Huang <i>et al.</i> , 2017).....	19
4.3 Network architecture of SSD .....	20
5.1 Bounding box evaluation: The right part of the figure shows how to calculate Intersection over Union (IoU), where as the left part shows two predicted boxes in Cyan color, the ground truth boxes in black color and their IoU values.....	23



Figure	Page	
5.2	Figure (a) shows two predicted boxes which are counted as True positive samples because the IoU value is $\geq 0.5$ . Figure (b) shows a predicted box which is counted as False positive sample because the IoU value is $< 0.5$ . Figure (c) shows a ground truth box which is considered as a False Negative sample because there is no True Positive prediction for the ground truth box. . . . .	24
5.3	The correlation between precision and recall with varying prediction score. . . . .	26
5.4	Precision vs Recall curve for different classes, evaluated with Faster R-CNN model. . . . .	27
5.5	Figure (a) and (b) show confusion matrices evaluated by <b>Faster R-CNN model</b> . (TP = True Positive, FN = False Negative, FP = False Positive, TN = True Negative) . . . . .	28
5.6	Figure (a) and (b) show confusion matrices evaluated by <b>SSD model</b> . (TP = True Positive, FN = False Negative, FP = False Positive, TN = True Negative) . . . . .	29
5.7	Shows the ROC curve for Faster R-CNN model . . . . .	30
5.8	Shows the ROC curve for SSD model . . . . .	31
5.9	Shows the AP (Average Precision) for each class and the overall mAP (mean Average Precision) value as the epoch number on the X-axis for Faster R-CNN model . . . . .	32
5.10	Shows the AP (Average Precision) for each class and the overall mAP (mean Average Precision) value as the epoch number on the X-axis for SSD model . . . . .	33

Figure	Page
5.11 Side-by-side comparisons of detections of the models Faster R-CNN and SSD.....	34
5.12 Side-by-side comparisons of detections of the models Faster R-CNN and SSD.....	35
5.13 Side-by-side comparisons of detections of the models Faster R-CNN and SSD.....	35
5.14 Side-by-side comparisons of detections of the models Faster R-CNN and SSD.....	36
5.15 Side-by-side comparisons of detections of the models Faster R-CNN and SSD.....	37
5.16 Side-by-side comparisons of detections of the models Faster R-CNN and SSD.....	37
6.1 Sample images with illegible handwriting.....	40
6.2 A sample handwritten document from the IAMonDo-database.....	40

## Chapter 1

### INTRODUCTION

Handwritten documents have gained popularity in domains such as education and business. The Formative Assessment with Computational Technologies (FACT) research team built an educational software which takes handwritten and text input. FACT research project is helping middle school math teachers understand how students are collaborating or working individually on a set of math problems (VanLehn *et al.*, 2016). FACT system analyzes the content written by the student and sends notifications (alerts) to the teacher regarding student performance. However, analyzing handwritten content is a challenging task. Content detection helps in the analysis by differentiating various types of content on a document.

A key task in analyzing a complex document is distinguishing between various content types such as text, math, graphics, table and so on. This differentiation facilitates the performance of specific recognition tasks depending on the content type (Indermühle *et al.*, 2010a; Okun *et al.*, 1999). My research goal is to detect interesting aspects of a handwritten document and assign each aspect a meaningful label. For example, one such aspect could be a region on the notes with a mathematical expression; in this case the label would be math. We call this task as content detection.

FACT system provides handwriting and text input without any restrictions. This means that the student can write calculations or draw diagrams on the document to come up with a solution. Sketch recognition, being a hard problem (Cheema and LaViola, 2012), a recognizer may find it difficult to correctly recognize the letters/equations/ diagrams on the whole document together. We hypothesize that content detection helps the recognizer perform better by suggesting which kind of information

to look for within each detected block; whether to look for text, math, diagram and so on.

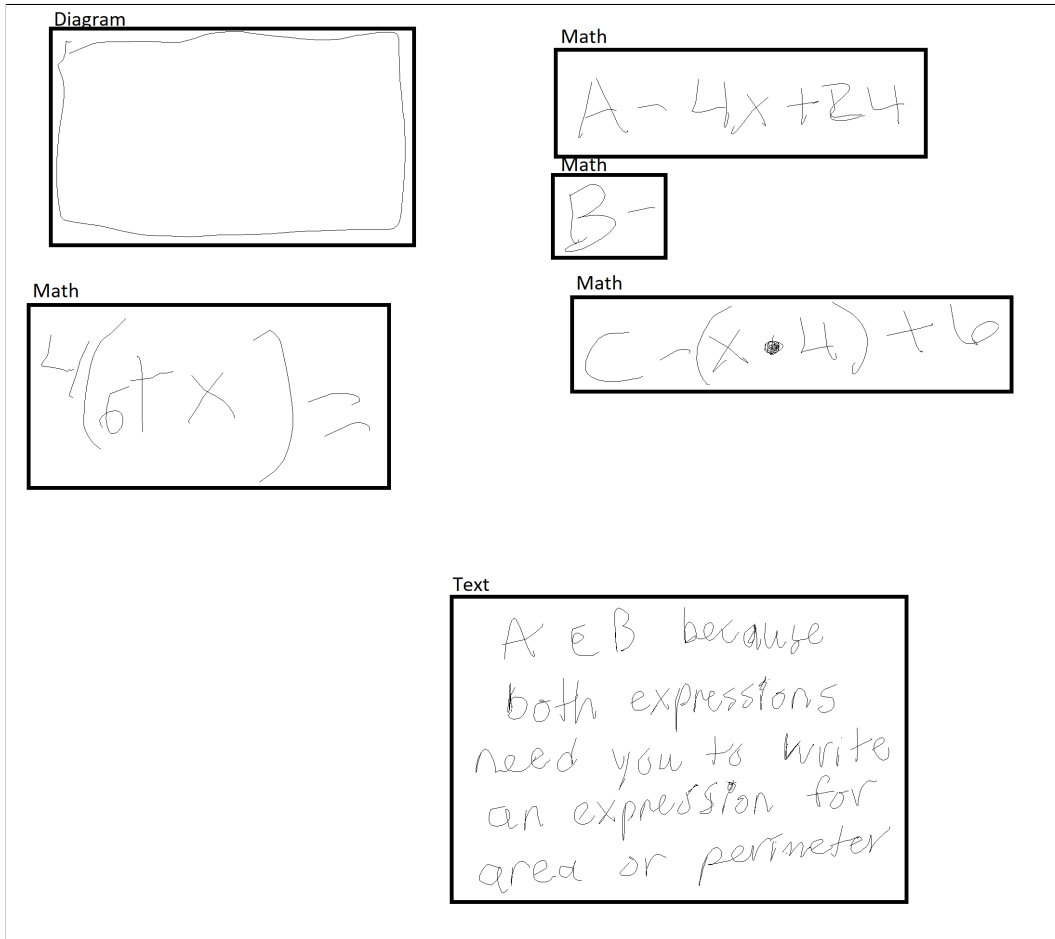


Figure 1.1: A training handwritten sample of a solved math problem from FACT project

A preliminary task in analyzing a handwritten document is to detect regions containing homogeneous pieces of data. In this context, a homogeneous piece of data is a group of handwritten strokes representing similar content. Different content types are text, math, diagram, cross out, table, graph, tick mark, arrow, and doodle (Table 3.1). For example, Figure 1.1 depicts different content types on a document. The Figure 1.1 shows a handwritten document with labelled rectangular boxes. The

handwritten document, also called a poster, has a math problem which has been solved by a student. We took a screen-shot of the handwritten data of a solved poster from FACT and labelled meaningful parts of the solution as shown in Figure 1.1. This image when processed by our trained model resulted with detections shown in Figure 1.2.

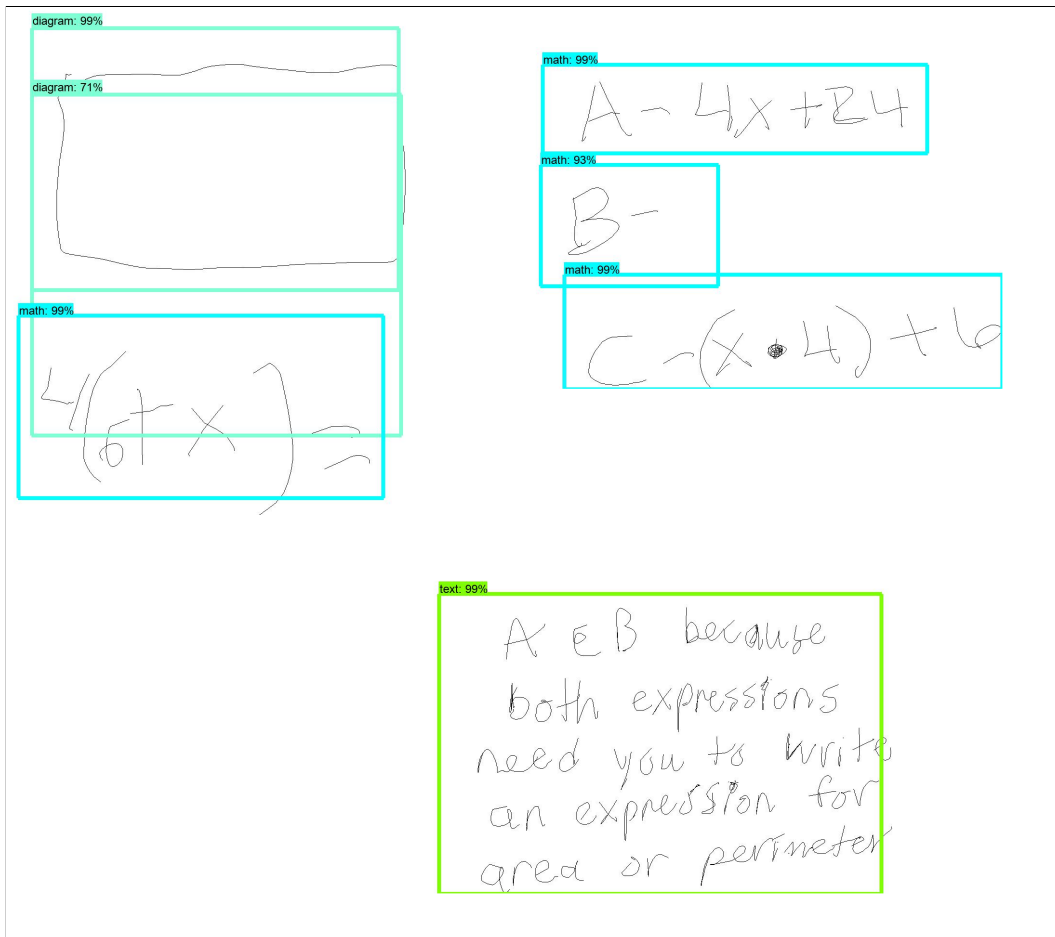


Figure 1.2: A detected handwritten sample of a solved math problem from FACT project

Recently, object detection techniques based on deep neural networks have been shown to perform well on natural images. Our research question is

**How accurately can we detect content type in a handwritten document**

**using object detection techniques.**

To investigate this question, we extracted examples of student handwritten work from the FACT system as images and labelled them individually. Additionally, this thesis required a good understanding of FACT system’s code base for the extraction of data. This thesis also required a thorough understanding of neural networks as the models in use are novel state-of-art deep networks with most of the code written in Python using the Tensorflow library.

We analyzed our dataset on two popular object detection techniques Faster Region-based Convolution Neural Network (Ren *et al.*, 2015) and Single Shot Detector (Liu *et al.*, 2016), with mean Average Precision (mAP) as the evaluation metric, a common evaluation metric for object detection tasks (Ren *et al.*, 2015; Everingham *et al.*, 2015).

## 1.1 Terminology

We use the following terminology in this document. In a handwritten document, we call the process of detecting meaningful rectangular boxes and assigning labels to them as ‘content detection’. In this document, the word ‘object detection’ is used more generally, on any kind of image, while ‘content detection’, on the other hand, is on images of handwritten student work. In this document, the word tagging is used interchangeably with labelling.

## 1.2 Reader’s Guide

This document is outlined in the following manner. Chapter 2 discusses different approaches to content detection on documents and object detection in general. The last section of chapter 2, *Preliminary Data Collection & Research*, describes a proof of concept done for the proposal of this thesis. Chapter 3 discusses how the handwritten

data was collected and gives a detailed description of the data and how it was labelled. In chapter 4, we give a brief introduction of the two neural networks Faster R-CNN and SSD. Chapter 5 demonstrates how the models were evaluated in detail. In chapter 6, we discuss some possible improvements. We conclude what we have done in chapter 7.

## Chapter 2

### RELATED WORK

A top-down approach for content detection is to detect interesting parts of the document as bounding boxes and assign a label to each bounding box using a classifier (Indermühle *et al.*, 2010a; Keysers *et al.*, 2007; Wong *et al.*, 1982). A bottom-up approach would be - to classify each stroke as a specific content type and then cluster similar strokes together, forming segments (Stahovich and Lin, 2016; Indermühle *et al.*, 2010a; Jain *et al.*, 2001; Strouthopoulos and Papamarkos, 1998). Both these approaches have been applied on digital ink data and on images of the documents as well. With the recent advancement of fast and accurate object detection techniques using convolutional neural networks (Huang *et al.*, 2017; Redmon *et al.*, 2016; Ren *et al.*, 2015), content detection is being done on images which contain variety of content. This thesis focuses on using recent object detection techniques to identify content present on images of handwritten documents.

Extracting semantic information from digital ink data is a challenging problem. The difficulty lies in detecting what type of drawings (with their boundaries) are present on the document. The detection increases the recognition accuracy for the upcoming tasks such as textual, shape and math recognition (Stahovich and Lin, 2016).

Content detection (Kleinberg *et al.*, 1998) remains a challenging problem, especially when it comes to differentiating text and math strokes. Traditionally, content detection was done using digital ink information or using pixel information of images (Fu and Kara, 2011). Content detection on images of handwritten documents and object detection on natural images (Redmon *et al.*, 2016; Ren *et al.*, 2015) are similar



in methodology. These object detection techniques have found a significant success in related fields. This thesis focuses on content detection using these recent object-detection deep neural network models on images of handwritten documents. We also discuss the challenges faced in collecting and labelling the data.

In general, object detection in images has a wide range of applications such as detection of traffic signs for autonomous cars (Escalera *et al.*, 1997), human action detection (Yu and Yuan, 2015), counting number of people in a very crowded scene, and estimating the number of vehicles in a traffic congestion (Onoro-Rubio and López-Sastre, 2016). With the recent advancement of neural networks, the object detection task has achieved better performance, both in speed and accuracy (Huang *et al.*, 2017). (Huang *et al.*, 2017) has a good review of different CNN models used for object detection.

## 2.1 Preliminary Data Collection & Research

This section describes a initial proof of concept that was done for the proposal of this thesis. The following paragraphs demonstrate a small experiment that was carried out for evidence.

Given any arbitrary math problem, students typically write text, equations, tables or graphs. Figure 2.1 shows a screen-shot of a solved math problem and the labels assigned to each meaningful portion of the image. The labelling was done by selecting rectangular regions on the image and assigning a label to them. We trained a DNN model proposed by (Ren *et al.*, 2015) with 21 labelled images and found high accuracy on 6 test images. Figure 2.2 shows the output showing the label and confidence percentage on the top left corner of each detected bounding box.

This proof of concept was performed using a DNN model from the Object detection library of Tensorflow (Huang *et al.*, 2017). The DNN model is known as faster R-CNN

model as proposed by (Ren *et al.*, 2015) and has been shown to give good results on *PASCAL Visual Object Classes* dataset (Everingham *et al.*, 2010), *Microsoft Common Objects in Context* dataset (Lin *et al.*, 2014), and others<sup>1</sup>. The faster R-CNN model is a state-of-art object detection neural network for performing fast object detection on images and has achieved better accuracy compared to most other models on both the PASCAL VOC and MS COCO datasets (Huang *et al.*, 2017).

---


<sup>1</sup>[https://en.wikipedia.org/wiki/List\\_of\\_datasets\\_for\\_machine\\_learning\\_research#Object\\_detection\\_and\\_recognition](https://en.wikipedia.org/wiki/List_of_datasets_for_machine_learning_research#Object_detection_and_recognition)

**Making and selling candles**

1. A student wants to earn some money by making and selling candles. He plans to sell 50. The production cost for each candle is \$4. Suppose that he can make 20 candles from each kit, and that each candle will sell for \$4.

Write the values for  $k$ ,  $n$ , and  $s$  in the boxes.

- The cost of buying the kit
- The number of candles that can be made with each kit
- The price at which he sells each candle
- The total profit made by the student



50	profit
60	price
4	cost
190	total

[2.a] How can you calculate the profit  $p$  using the given values of  $k$ ,  $n$ , and  $s$ ?

$p = ns - k$

[2.b] Would your methods change if the values of  $k$ ,  $n$ , and  $s$  were different? Explain your answer.

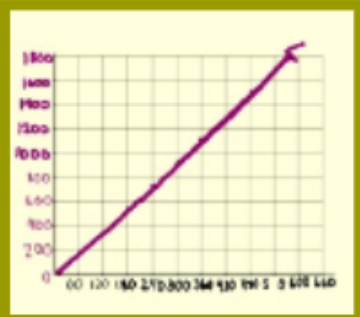
No, it wouldn't change because the values don't change. It depends on what your factor is, such as the cost or profit total.

[3] Now that you know the profit, go back and erase the selling price of each candle,  $s$ . Suppose you didn't know  $s$ . How could you figure it out?

We could put it into an equation like  $50 + 60s = 190$

$p = ns - k$

$n$	$p$
60	190
120	380
180	570
240	760



Legend

- text
- image
- math
- table
- graph

5. Write equations for each variable showing the relationship between the variables.

$p = ns - k$	$k = \frac{p - ns}{-1}$
$n = \frac{p + k}{s}$	$s = \frac{p + k}{n - p}$

$k = -$

Figure 2.1: A labelled training sample of a solved math problem from FACT project

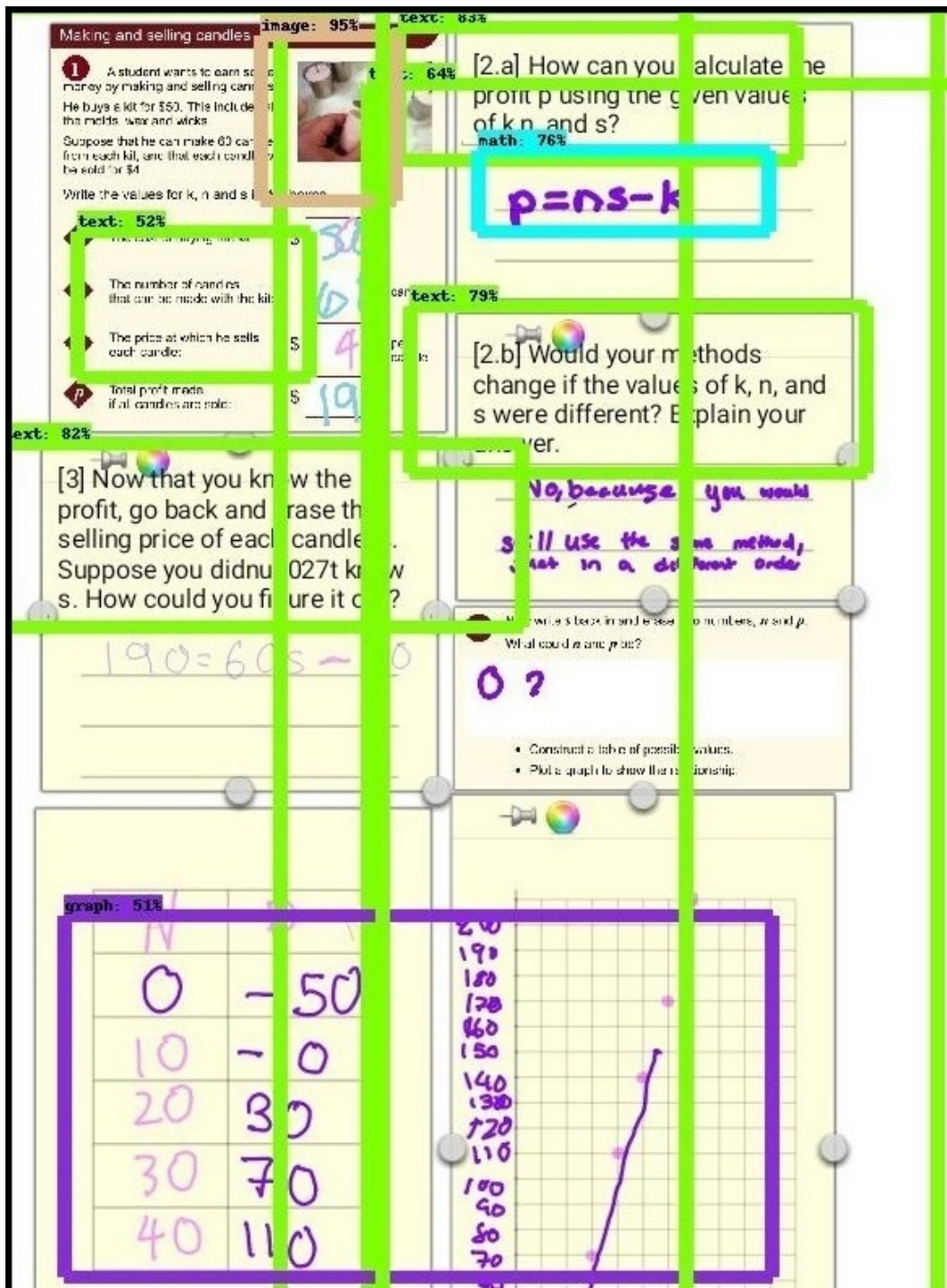


Figure 2.2: A test image of a solved math problem from FACT project detected by the neural network model

## Chapter 3

### DATA COLLECTION

#### 3.1 Methods

By organizing 32 classroom trials in England and California, the FACT project has collected more than 4049 unique examples of handwritten student work on complex math problems (VanLehn *et al.*, 2016). Figure 3.1 shows a handwritten solution also called as a poster, contains anything from handwriting, text, and images. Students could write calculations or draw diagrams on the poster to come up with a solution. These solutions are written by middle school students in a classroom setting.

The first step in processing the data was to extract only the handwritten strokes from the JSON-formatted poster. The removal of typed text and math was necessary because the system was already aware of its presence on the poster. The handwritten strokes were then further separated by their location on the poster - whether they were on a card or on the canvas itself. All the strokes that were present on the canvas were drawn on a Graphics2D (`java.awt.Graphics2D`) object and then rendered as an image. A similar approach was followed for the strokes present on cards, which were extracted separately as individual images per card. On both the canvas and cards, if there were no or few strokes<sup>1</sup>, they were discarded. Algorithm 1 was used to extract handwritten strokes. Figures 3.2 and 3.3 show samples of extracted strokes.

---

<sup>1</sup>2 to 3 strokes which do not make text, math or shape

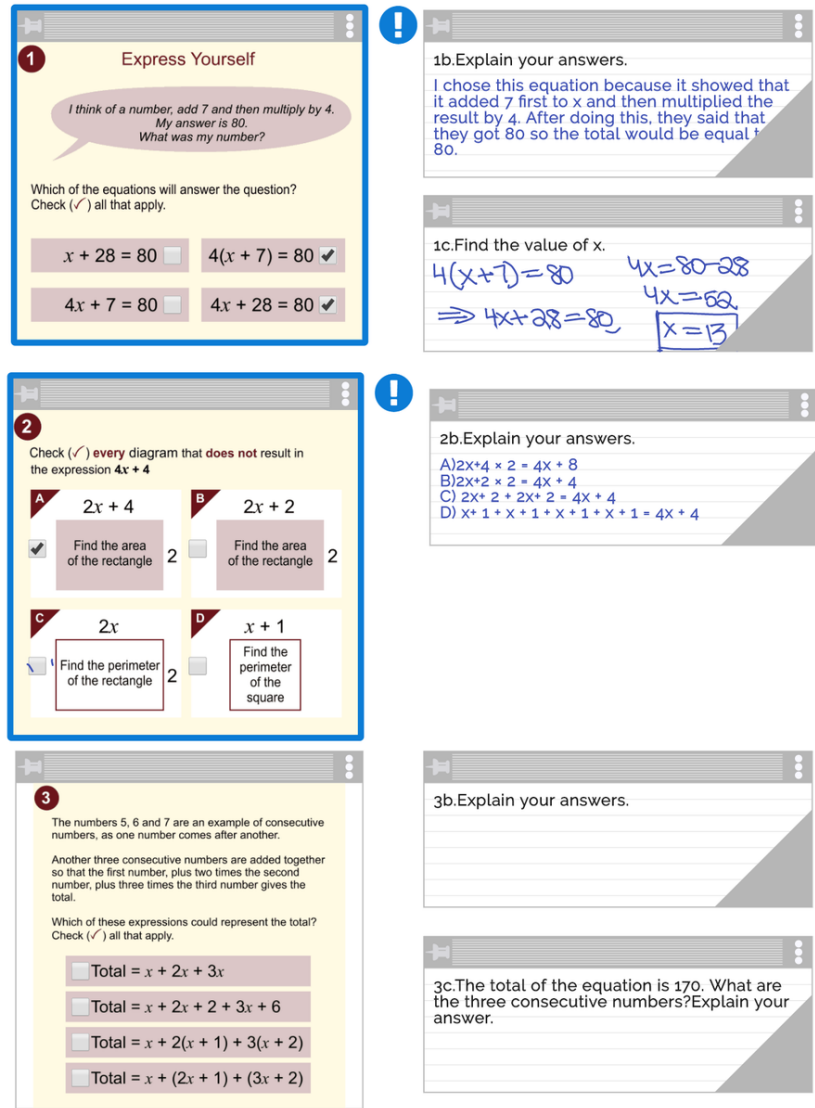


Figure 3.1: A sample solved problem from FACT project

---

**Algorithm 1** How to render an image from strokes

---

**Input** : strokesList: List of Strokes

**Output:** Image

BufferedImage bi = new BufferedImage();

Graphics2D g = bi.createGraphics();

**while** there exist a next stroke in strokesList **do**

    Stroke s = strokeList.next()

**for**  $i=0 ; i < s.size()-1 ; i++$  **do**

            Point p1 = s.getPoint(i);

            Point p2 = s.getPoint(i+1);

            g.draw(new Line2D.Double(p1.getX(), p1.getY(), p2.getX(), p2.getY()));

**end**

**end**

ImageIO.write(buffer, "png", outputImageFile);

---

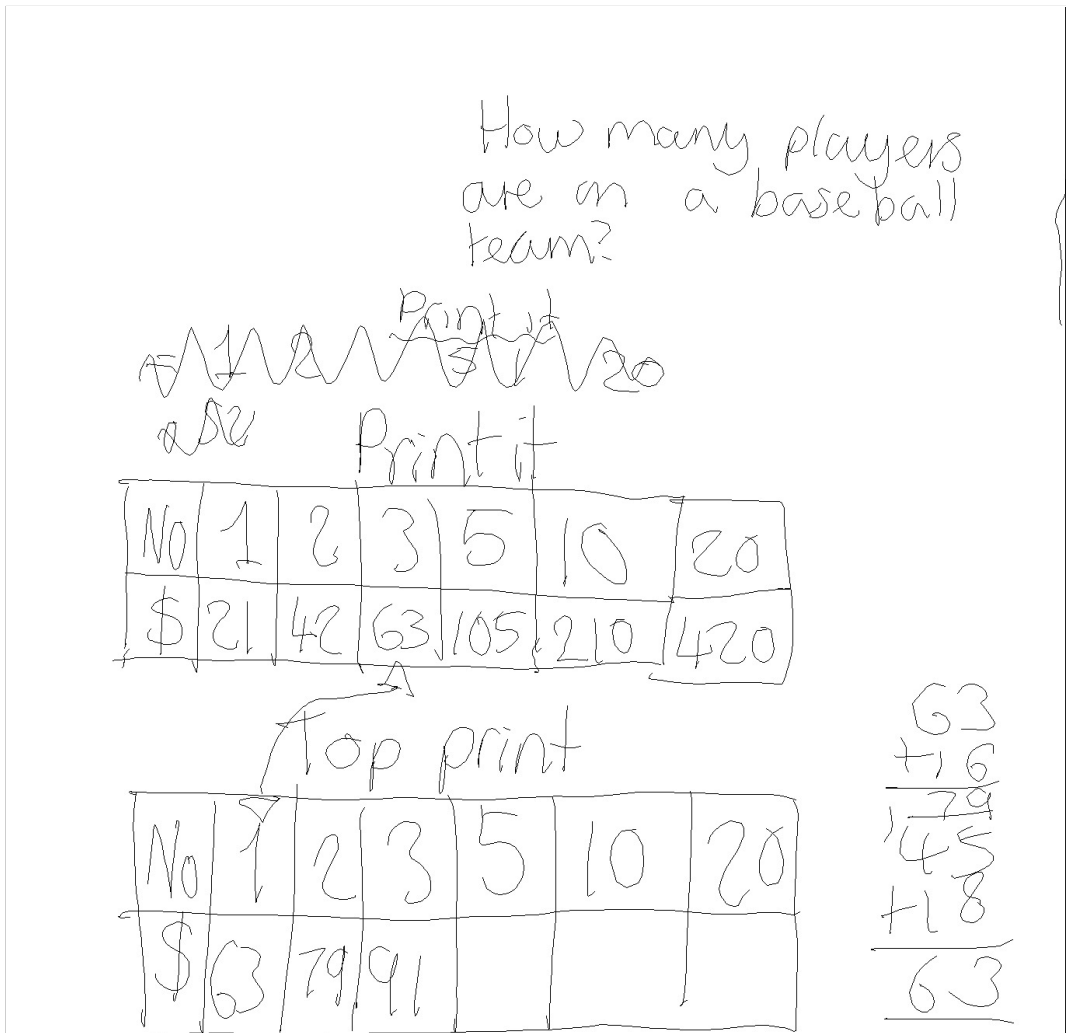


Figure 3.2: A sample image of a poster containing only strokes

### 3.2 Tagging

The handwritten images had to be tagged before using them for training purposes. They were tagged using the Microsoft's Visual Object Tagging Tool (VoTT). Tagging involves selecting a rectangular region on the photo and giving it a label. A meaningful label could be text, math, diagram, cross out, table, graph, tick mark, arrow or doodle (Table 3.1).

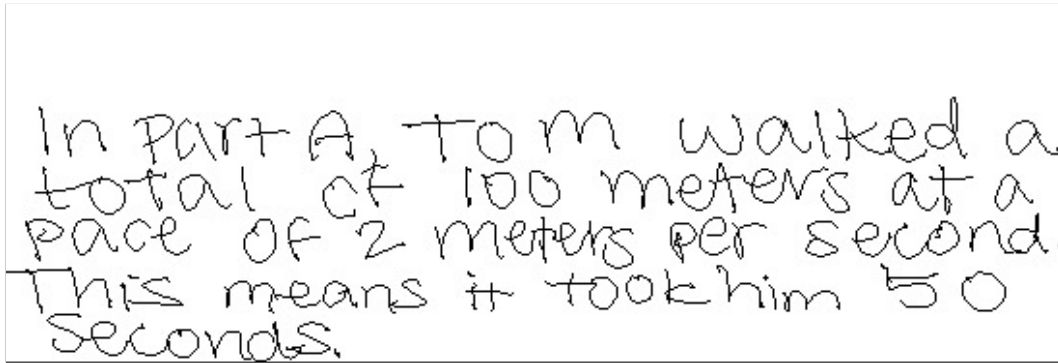


Figure 3.3: A sample image of a card containing only strokes

Since Microsoft's VoTT<sup>2</sup> was an easy-to-use software with a portable executable chosen for image tagging. The tagging process can be resumed if there was a pause in between. The labels are stored in a format suitable for the neural network model, once the tagging is done. Adversely, there is one known issue with the VoTT software - once tagged, the rectangular boxes' size and position change when revisiting an image. A work around for this issue is to rearrange the boxes when revisiting an image. We revisited every image in our dataset and rearranged the rectangular box's size and position. This assured that the tagging was properly done, but it is recommended to use a different tagging tool or to use VoTT after this issue gets resolved.

### 3.3 Data Statistics

A total of 4049 documents were extracted in the form of images. 90% of these form the training set and 10% form validation set. All the images were labelled using VoTT software with tags being text, math, diagram, cross out, table, graph, tick mark, arrow, and doodle. Figure 3.4 shows the number of labels of each class in our dataset. We have more examples of text, math, diagram and doodle and less examples of cross-out, table, graph, tick mark and arrow.

---

<sup>2</sup><https://github.com/Microsoft/VoTT>



Label	Description
Text	A piece of English writing on the notes, either handwritten or typed
Math	A segment on the notes with mathematical character(s)
Diagram	Geometrical shapes
Cross-out	Anything on the notes that has been crossed out
Table	Tabular information about variable(s)
Graph	A plot showing the relation between two variables
Tick mark	A check mark written with hand
Arrow	Arrow showing a direction or position
Doodle	A drawing which is not relevant to the assigned work

Table 3.1: Labels given to a region on a handwritten notes

Number of Labels	Dataset	
	Training set	Validation set
Text	503	116
Math	571	106
Diagram	103	26
Cross-out	31	3
Table	27	2
Graph	6	0
Tick mark	16	0
Arrow	73	7
Doodle	134	22
<b>Total</b>	<b>1464</b>	<b>282</b>

Figure 3.4: shows the number of labels of each class in our dataset

### METHODOLOGY

In this section, we discuss the two models used for content detection. The first section gives a brief introduction to convolutional networks, and the next two sections discuss the models.

#### 4.1 Convolutional Neural Networks

Figure 4.1 shows a typical convolutional neural network model. As shown in the Figure 4.1, the model is divided into two phases, Feature learning and Classification. In the feature learning phase, the model extracts a feature vector from an input image. This phase, sometimes referred as feature extractor, generally, consists of convolution and pooling layers with activation functions like ReLU (Nair and Hinton, 2010). These layers reduce the image to a smaller size producing a feature vector as the output. An activation function is a mathematical function which defines the output given an input or a set of inputs. In the classification phase, the extracted feature vector is further reduced to generate a category.

We used meta-architectures (Huang *et al.*, 2017) of the initially proposed Faster R-CNN (Ren *et al.*, 2015) and SSD (Liu *et al.*, 2016). Meta-architectures differ from the originally proposed architectures (Faster R-CNN & SSD) in a way that we can plug-in different feature extractors and not the one's originally used. There are various features extractors specifically for image data. Some of them are VGG model ( named after the Visual Geometric Group (Simonyan and Zisserman, 2014)), Resnet (Residual Network, (He *et al.*, 2016)) and inception network (Szegedy *et al.*, 2015). These are convolutional networks that are proven to produce good results on image

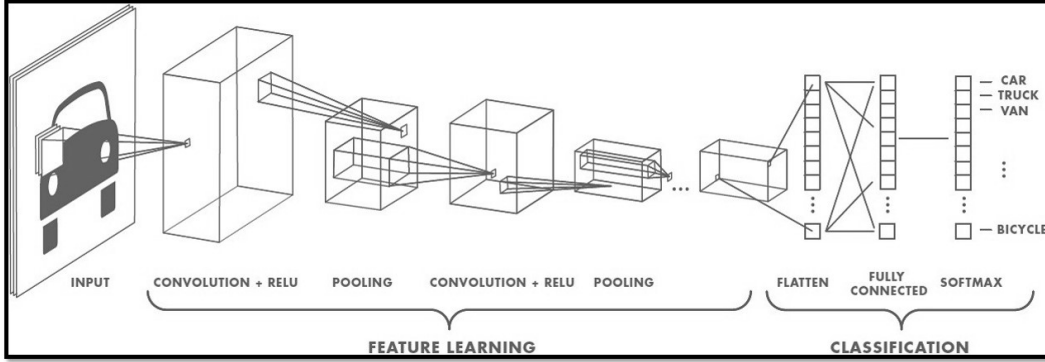


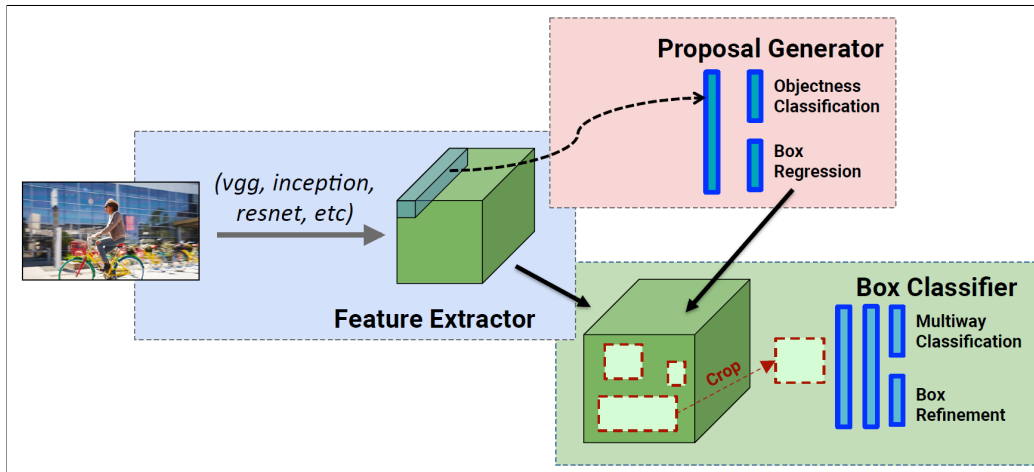
Figure 4.1: A typical supervised convolutional neural network architecture showing feature learning and classification phases

data. The first few layers in these feature extractors are known to detect cusps and curves. The deeper layers detect shapes and bodies.

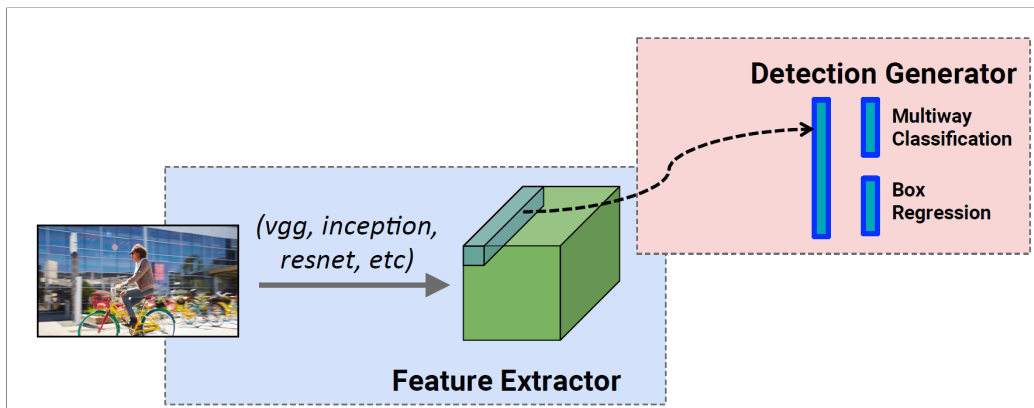
## 4.2 Faster R-CNN

As described (Huang *et al.*, 2017), the model used is a meta-architecture of the Faster R-CNN (Region-based Convolutional Neural Network). The model, Faster R-CNN (Ren *et al.*, 2015) was implemented to improve the speed of object detection while not compromising the accuracy. The models - Fast R-CNN (Girshick, 2015) and R-CNN (Girshick *et al.*, 2014) were released before Faster R-CNN was released. While all three of them share a similar network architecture, the later one's improvised on the speed.

As shown in Figure 4.2a, Faster R-CNN has two convolutional networks. The first network predicts 300 bounding boxes with their box regressions. Box regressions are small corrections to refine the predicted bounding box. The network also provides an *objectness score* which determines how confident the model is about the existence on object. Note that the model doesn't predict the class label for each region yet. A second network determines the classes for each proposal and also predicts a box



(a) Faster R-CNN



(b) Single Shot Detector (SSD)

Figure 4.2: Architectural comparison of the two models. (a) Faster R-CNN has two networks - one to generate region proposals, the other to predict a class to find a box refinement for each proposal. (b) SSD has a single network to find regions, classify them and to refine to each region proposal. Images are from (Huang *et al.*, 2017)

refinement by taking the region proposals and the features extracted from the first network. Note that the second network works as an image classifier and thus can be a pre-trained network trained on different set of data. The Faster R-CCN model we used, has a convolutional network (also called as feature extractor), is known as ResNet-101 (residual network) as implemented in the paper (He *et al.*, 2016).

### 4.3 Single Shot Detector (SSD)

As described (Huang *et al.*, 2017), the model used is a meta-architecture of the Single Shot Detector (SSD). As shown in Figure 4.2b, SSD has just one feature extractor (CNN) to predict, refine, and classify the region proposals. The network also provides a *prediction score* which determines how confident the model is about the class.

The SSD model we used, has a feature extractor, is known as inception network as implemented in (Szegedy *et al.*, 2015). The neural network architecture of SSD model is shown in the figure 4.3.

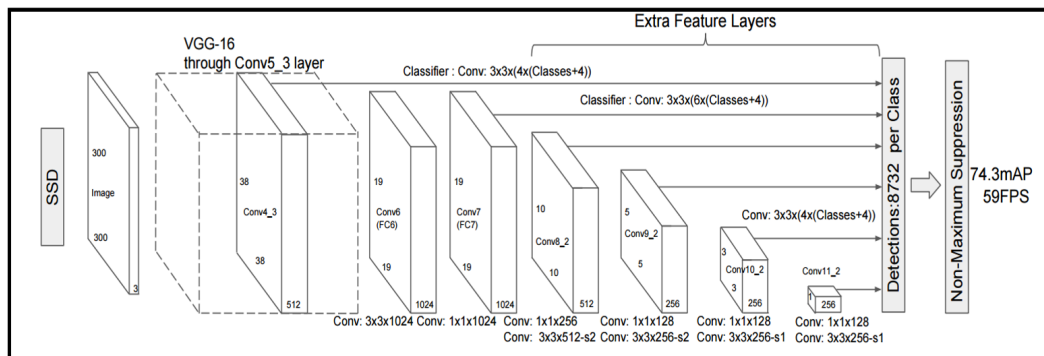


Figure 4.3: Network architecture of SSD

#### 4.4 System Specifications

Training of the model - Faster R-CNN, was done on a “Dell Inspiron 15.6 Laptop with Intel Core i5, 8GB Memory and NVIDIA GeForce GTX 1050 Ti”. Training of the SSD model was done on a remote machine with “Nvidia Tesla K40m (12GB RAM)”. The proof of concept mentioned in section 2.1 was performed on a system with Core i7 processor and 32GB RAM (and no GPU).

## Chapter 5

### RESULTS

In this section, we define and explain the evaluation metrics of this content detection task. We analyze both speed and accuracy in our experiments.

#### 5.1 Intersection over Union (IoU)

Given a test image, our trained content detection model outputs a set of predicted bounding boxes with their labels and prediction score.

Each predicted bounding box is evaluated based on how much it overlaps with the ground truth object. We used a bounding box evaluation metric called Intersection over Union (IoU). Each detected bounding box is assigned to ground truth object and judged to be true/false positive by measuring the bounding box overlap (Everingham *et al.*, 2010). The IoU value is the proportion of intersection of the bounding box and the predicted box to their union. Figure 5.1 shows how we calculate IoU value for a given bounding box prediction.

Figure 5.2 shows samples of true positive, false positive and false negative. If the model correctly labels the text with an IoU value greater than 0.5, it is considered a true positive. If the IoU value is less than 0.5 or if the model incorrectly predicts a bounding box when no such ground truth object is present, the prediction will be regarded as false positive. For example, if there is no text present but the model predicts a bounding box as math or text, then that predicted box will fall into the false positive category. A false negative will occur when there is a ground truth object and the model failed to identify it. A true negative does not occur in object detection because if there is no ground truth and there is no prediction, then there is nothing



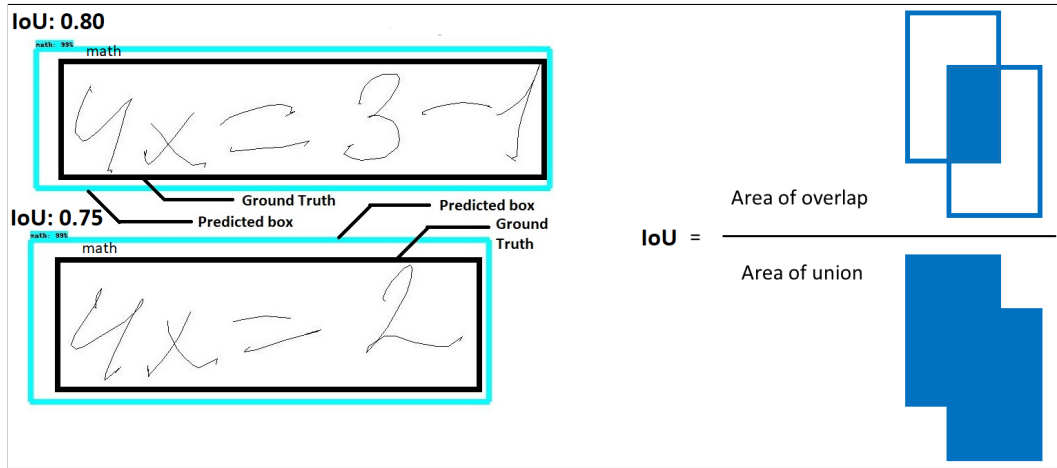


Figure 5.1: Bounding box evaluation: The right part of the figure shows how to calculate Intersection over Union (IoU), where as the left part shows two predicted boxes in Cyan color, the ground truth boxes in black color and their IoU values.

to measure.

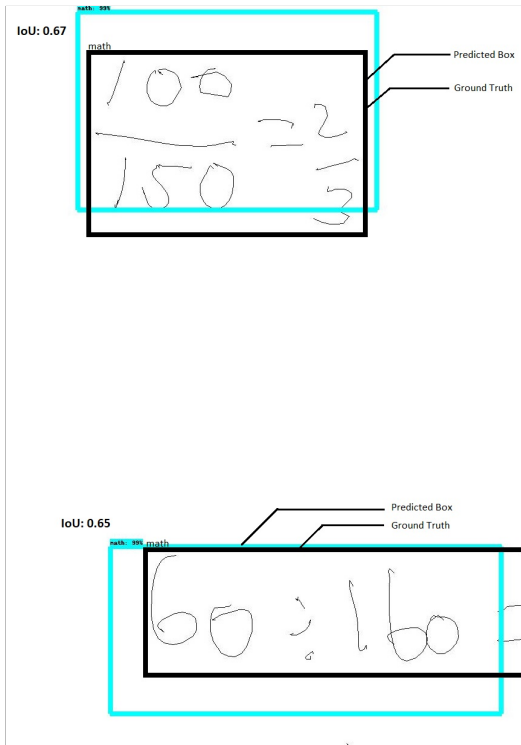
## 5.2 Precision and Recall

Precision is defined as the proportion of all predictions which are from the positive class i.e. is calculated by dividing the number of true positives with the total number of true positives and false positives. The recall value is calculated by dividing the number of true positives by the number of true positives and false negatives. To calculate IoU value, precision and recall; only bounding boxes and their predicted classes were used but not the prediction score.

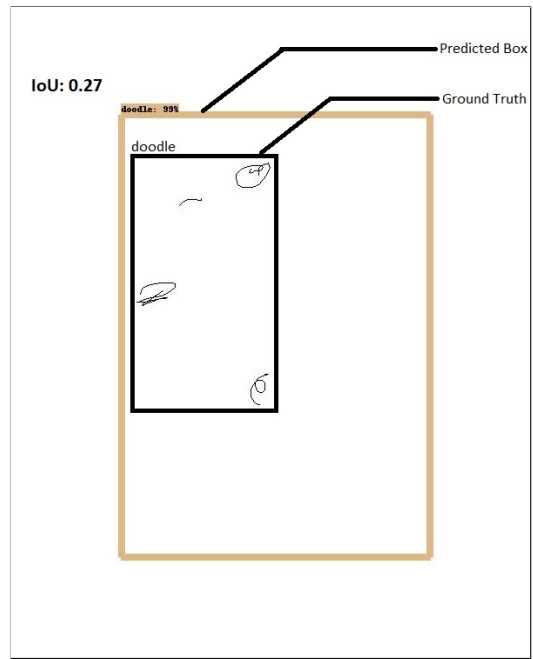
$$\text{precision} = \frac{\text{truepositives}}{\text{truepositives} + \text{falsepositives}} \quad (5.1)$$

$$\text{recall} = \frac{\text{truepositives}}{\text{truepositives} + \text{falsenegatives}} \quad (5.2)$$

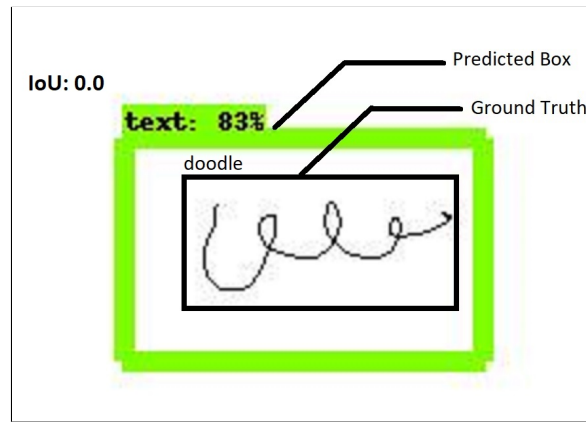
The number of True/False positives is limited by the prediction score of the bounding box. The prediction score gives the confidence level for the detected bounding



(a) True Positive samples



(b) False Positive sample



(c) False Negative sample

Figure 5.2: Figure (a) shows two predicted boxes which are counted as True positive samples because the IoU value is  $\geq 0.5$ . Figure (b) shows a predicted box which is counted as False positive sample because the IoU value is  $< 0.5$ . Figure (c) shows a ground truth box which is considered as a False Negative sample because there is no True Positive prediction for the ground truth box.

Prediction score	Precision	Recall
1.0	1.0	0.0
0.8	0.7	0.6
0.7	0.7	0.6
0.6	0.6	0.6
0.5	0.6	0.7
0.4	0.6	0.7
0.3	0.5	0.7
0.2	0.5	0.7
0.1	0.5	0.7
0.0	0.0	0.9

Table 5.1: The correlation between precision and recall with varying prediction score

box, so a prediction score of 1.0 means that the model is very confident about the predicted class and the possibility of the corresponding bounding box to be a true positive sample is highly likely. On the other hand, a prediction score of 0.0 means that the model is least confident about the predicted class, and the possibility of the corresponding bounding box to be a false negative is highly likely. Prediction score plays a defining role in the calculation of precision and recall. For example, if we were to cap the prediction score to 0.8 (which would mean any bounding box with a score less than 0.8 would not be taken into consideration), the number of false positives decrease and number of false negatives increase. Our results follow the common fact that the prediction score is directly proportional to precision and inversely proportional to recall.

Table 5.1 and the corresponding graph 5.3 show this correlation between precision

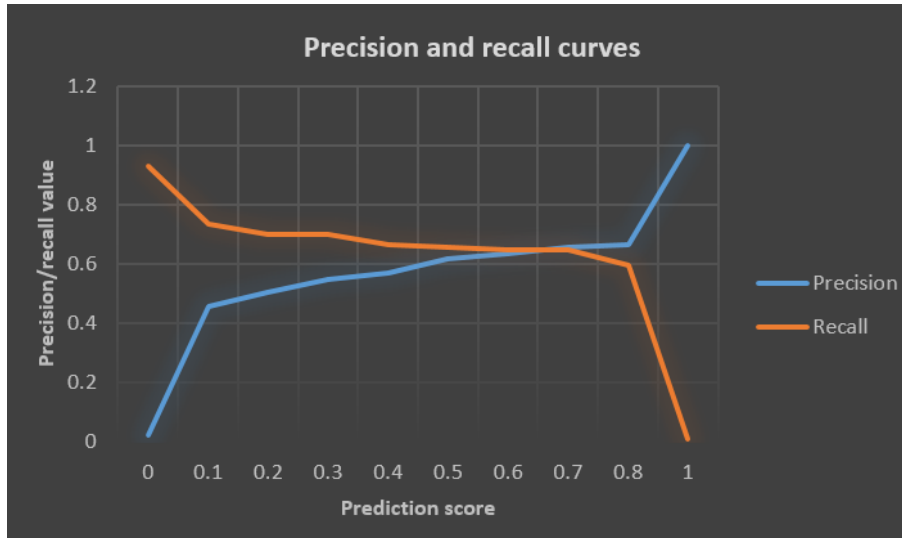


Figure 5.3: The correlation between precision and recall with varying prediction score.

and recall with varying prediction scores. Figure 5.4 shows the scatter plot of precision and recall for all the classes. These values are evaluated on the validation set with 1,400 epochs<sup>1</sup> of training on Faster R-CNN.

A confusion matrix is often used to describe the performance of a classification task (Bradley, 1997). Figures 5.5 and 5.6 show confusion matrices evaluated by the two models, evaluated at prediction score = 0.5.

### 5.3 ROC Curves

As described in (Hastie *et al.*, 2009) - “The receiver operating characteristic curve (ROC) is a commonly used summary for assessing the tradeoff between sensitivity (true positive rate) and specificity (false positive rate)”. It is a plot between true positive rate and false positive rate as we vary prediction score from 1 to 0.

True positive rate, recall, and sensitivity are all the same and is defined as number of true positives divided by number of positive samples. In our case, number of

<sup>1</sup>An epoch is one complete run of the training set to be learned on a model.

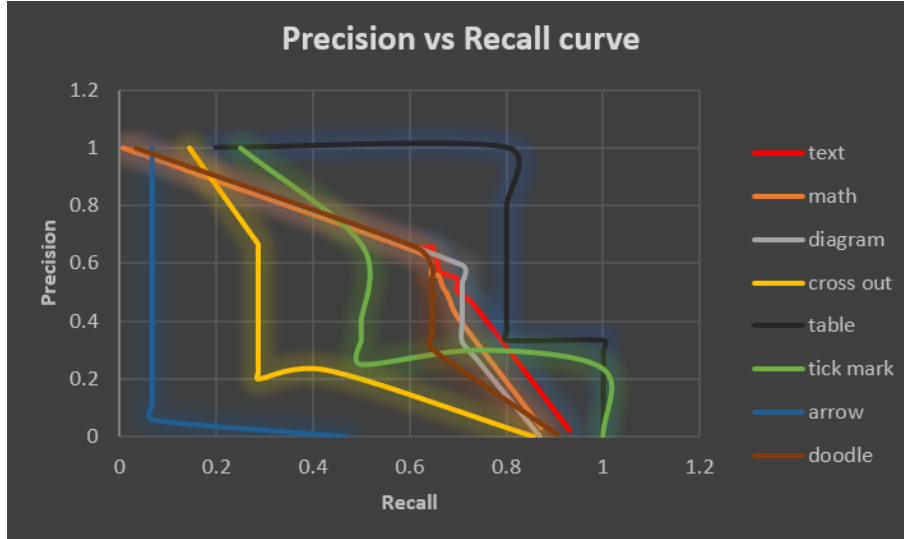


Figure 5.4: Precision vs Recall curve for different classes, evaluated with Faster R-CNN model.

positive samples is equal to the number of ground truth boxes.

$$True\ positive\ rate = recall = sensitivity = \frac{\sum true\ positives}{\sum positive\ samples} \quad (5.3)$$

False positive rate or Fall-out is defined as the number of false positives divided by the number of negative samples. In our case, number of negative samples is equal to total number of predictions made by the model minus number of true positives.

$$False\ positive\ rate = Fall - out = specificity = \frac{\sum false\ positives}{\sum negative\ samples} \quad (5.4)$$

Figures 5.7 and 5.8 shows the ROC curves for the models - Faster R-CNN and SSD respectively. We see that the ROC curves for the both the models look almost similar. The area under an ROC curve of a specific class signifies how well the classifier does compared to other classes (Bradley, 1997) (Hanley and McNeil, 1982). The dotted black line in Figures 5.7 and 5.8 shows the ROC curve for a random classifier. The

Class: Text		Predicted value (as predicted by the model)	
		positives	negatives
Actual Value	positives	TP = 106	FN = 10
	negatives	FP = 1388	TN = 3135

Class: Math		Predicted value (as predicted by the model)	
		positives	negatives
Actual Value	positives	TP = 2	FN = 30
	negatives	FP = 330	TN = 3426

Class: Diagram		Predicted value (as predicted by the model)	
		positives	negatives
Actual Value	positives	TP = 27	FN = 4
	negatives	FP = 5063	TN = 4903

Class: Cross out		Predicted value (as predicted by the model)	
		positives	negatives
Actual Value	positives	TP = 6	FN = 1
	negatives	FP = 1957	TN = 2998

(a) Confusion matrices for classes - Text, Math, Diagram, and Cross out

Class: Table		Predicted value (as predicted by the model)	
		positives	negatives
Actual Value	positives	TP = 5	FN = 0
	negatives	FP = 2110	TN = 1627

Class: Tick mark		Predicted value (as predicted by the model)	
		positives	negatives
Actual Value	positives	TP = 4	FN = 0
	negatives	FP = 5786	TN = 9864

Class: Arrow		Predicted value (as predicted by the model)	
		positives	negatives
Actual Value	positives	TP = 6	FN = 9
	negatives	FP = 1530	TN = 4370

Class: Doodle		Predicted value (as predicted by the model)	
		positives	negatives
Actual Value	positives	TP = 28	FN = 3
	negatives	FP = 2310	TN = 943

(b) Confusion matrices for classes - Table, Tick mark, Arrow, and Doodle

Figure 5.5: Figure (a) and (b) show confusion matrices evaluated by **Faster R-CNN model**. (TP = True Positive, FN = False Negative, FP = False Positive, TN = True Negative)

Class: Text		Predicted value (as predicted by the model)	
		positives	negatives
Actual Value	positives	TP = 68	FN = 33
	negatives	FP = 41	TN = 2956

Class: Math		Predicted value (as predicted by the model)	
		positives	negatives
Actual Value	positives	TP = 94	FN = 12
	negatives	FP = 5209	TN = 1319

Class: Diagram		Predicted value (as predicted by the model)	
		positives	negatives
Actual Value	positives	TP = 23	FN = 3
	negatives	FP = 309	TN = 282

Class: Cross out		Predicted value (as predicted by the model)	
		positives	negatives
Actual Value	positives	TP = 4	FN = 2
	negatives	FP = 64	TN = 761

(a) Confusion matrices for classes - Text, Math, Diagram, and Cross out

Class: Table		Predicted value (as predicted by the model)	
		positives	negatives
Actual Value	positives	TP = 5	FN = 0
	negatives	FP = 568	TN = 41

Class: Tick mark		Predicted value (as predicted by the model)	
		positives	negatives
Actual Value	positives	TP = 2	FN = 0
	negatives	FP = 40	TN = 434

Class: Arrow		Predicted value (as predicted by the model)	
		positives	negatives
Actual Value	positives	TP = 6	FN = 7
	negatives	FP = 315	TN = 740

Class: Doodle		Predicted value (as predicted by the model)	
		positives	negatives
Actual Value	positives	TP = 20	FN = 2
	negatives	FP = 2140	TN = 113

(b) Confusion matrices for classes - Table, Tick mark, Arrow, and Doodle

Figure 5.6: Figure (a) and (b) show confusion matrices evaluated by **SSD model**.  
(TP = True Positive, FN = False Negative, FP = False Positive, TN = True Negative)

area under the ROC curve of a random classifier is 0.5. It means that if a particular class has an area under ROC curve less than 0.5, then it is better to just randomly predict rather than using the model. We see that only the ‘arrow’ class under performs in both the models.

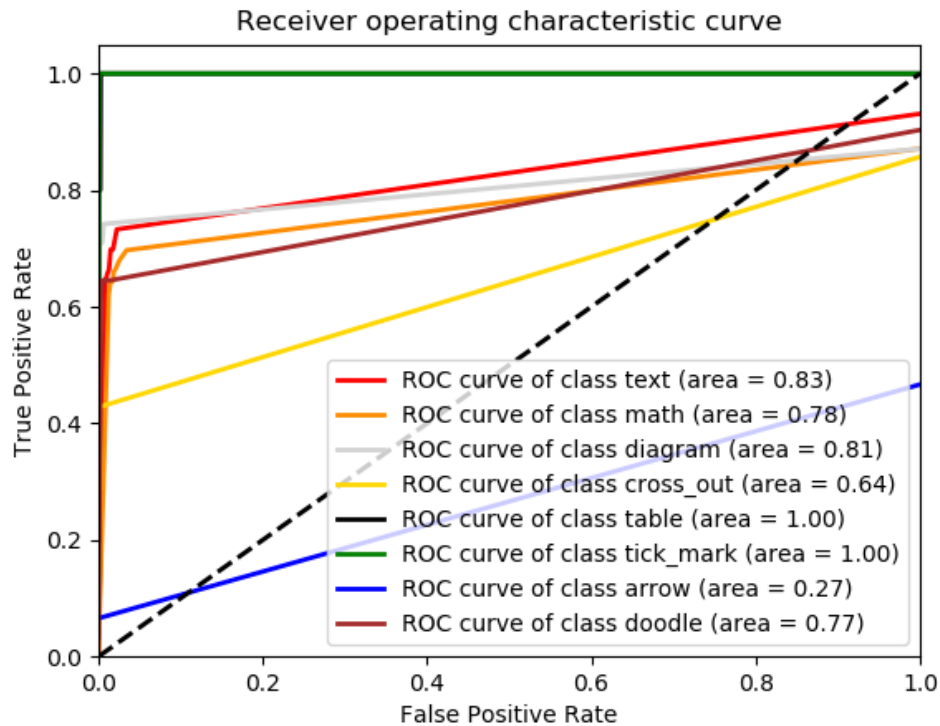


Figure 5.7: Shows the ROC curve for Faster R-CNN model

#### 5.4 Mean Average Precision (mAP)

The commonly used Mean Average Precision (mAP) for evaluating the quality of object detectors, is the mean of the per-class average precisions<sup>2</sup>. It is computed as per the protocol of the PASCAL VOC Challenge 2010-2012. Average Precision (AP)

<sup>2</sup>[https://github.com/tensorflow/models/blob/master/research/object\\_detection/g3doc/evaluation\\_protocols.md](https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/evaluation_protocols.md)



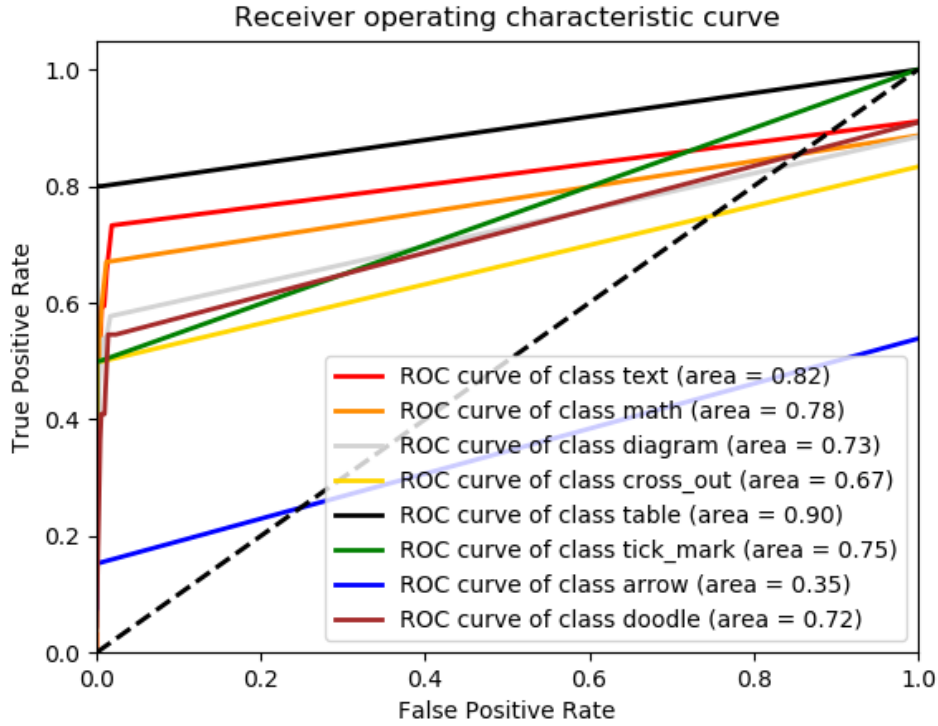


Figure 5.8: Shows the ROC curve for SSD model

is computed as the average of maximum precision at 11 recall levels as shown in the equation 5.5.  $AP_r(i)$  is the maximum precision for any recall values exceeding  $i$ .

$$AP = \frac{1}{11} \times (AP_r(0) + AP_r(0.1) + AP_r(0.2) + \dots + AP_r(1.0)) \quad (5.5)$$

$$AP_r(i) = \max_{j \geq i} Precision(j) \quad (5.6)$$

The mAP value depends on two factors: The Intersection over Union (IoU) number and the prediction score that the model provides for each predicted box. While averaging the precision values account for the varying prediction score, the IoU threshold of 0.5 was deliberately set low to account for the inaccuracies in bounding boxes in the ground truth data (Everingham *et al.*, 2010). We achieved a mAP of 55.3% and 54.5% on evaluation set with Faster R-CNN and SSD respectively. Figures 5.9

and 5.10 shows the Average Precision (AP) for each class and the overall mAP as the epoch number changes. On natural images (PASCAL VOC 2007 dataset), Faster R-CNN achieved 69.9% mAP and SSD achieved 74.3% mAP (Ren *et al.*, 2015; Liu *et al.*, 2016).

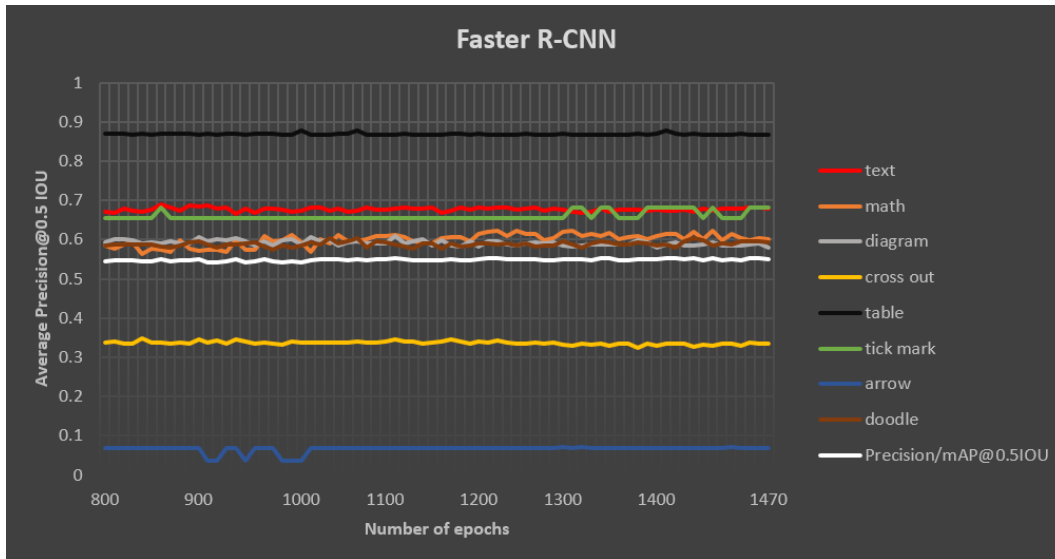


Figure 5.9: Shows the AP (Average Precision) for each class and the overall mAP (mean Average Precision) value as the epoch number on the X-axis for Faster R-CNN model

## 5.5 Time analysis

With Tensorflow, after a model has learned the weights, the result can be saved as a graph. The saved graph with all weights and variables takes 183MB and 55MB for models Faster R-CNN and SSD respectively. SSD has smaller network when compared to Faster R-CNN and thus loads and detects at a faster rate. Table 5.2 shows the time comparison for the two models when loaded on a Tesla K40m with 12GB memory.

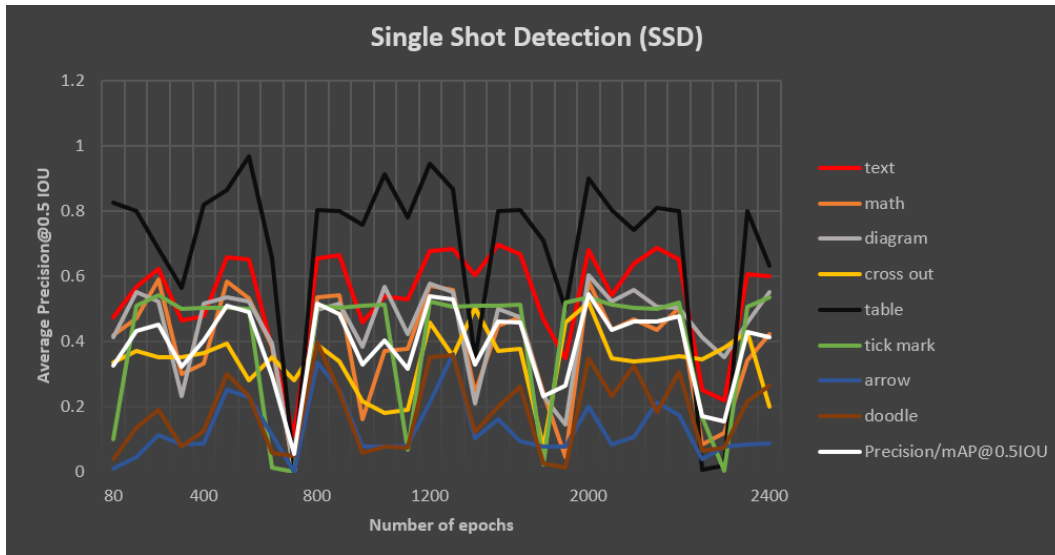


Figure 5.10: Shows the AP (Average Precision) for each class and the overall mAP (mean Average Precision) value as the epoch number on the X-axis for SSD model

Model	size on disk in Mega Bytes (MB)	time to load into memory on average	time to detect an image on average
Faster R-CNN	183 MB	2.03 seconds	2.83 seconds
SSD	56 MB	1.60 seconds	1.68 seconds

Table 5.2: Time comparison of the two models on Nvidia Tesla K40m (12GB)

## 5.6 Example detections

In Figures 5.11 to 5.16, we show side-by-side comparisons of some test-set detections of the models Faster R-CNN and SSD. We hand-tuned the threshold for prediction score as 0.5 for visual attractiveness.

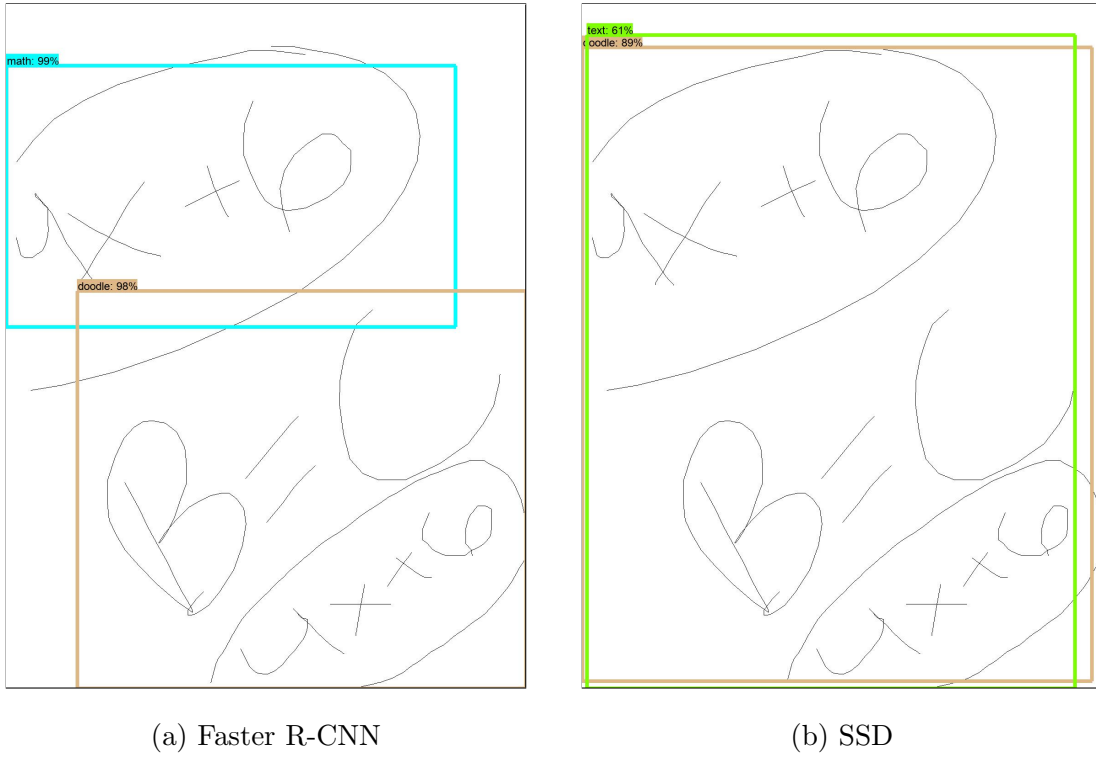
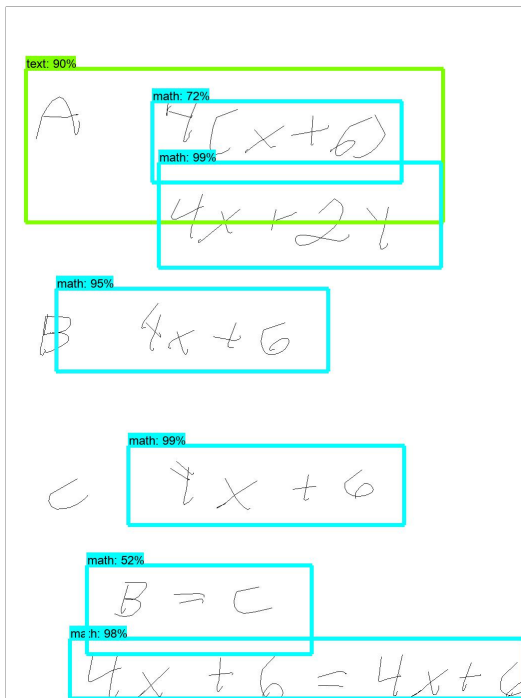
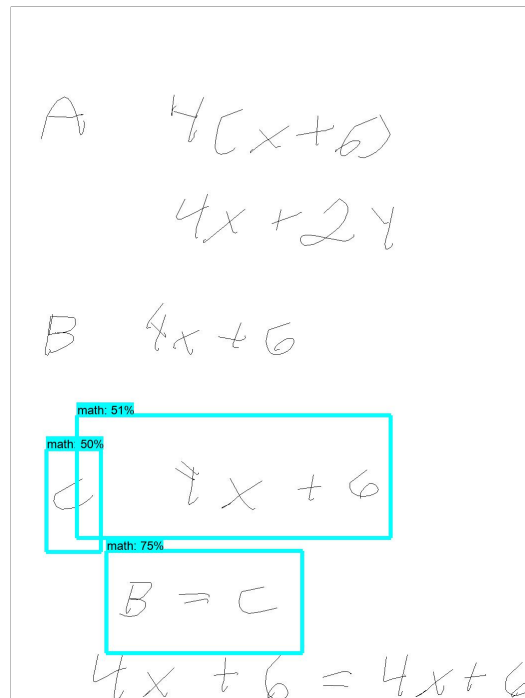


Figure 5.11: Side-by-side comparisons of detections of the models Faster R-CNN and SSD.

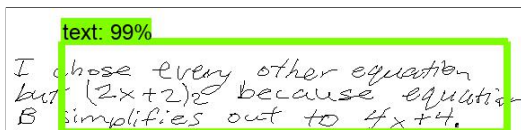


(a) Faster R-CNN

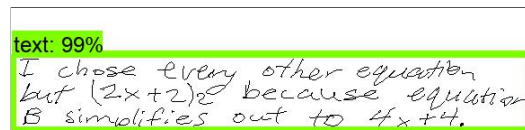


(b) SSD

Figure 5.12: Side-by-side comparisons of detections of the models Faster R-CNN and SSD.



(a) Faster R-CNN



(b) SSD

Figure 5.13: Side-by-side comparisons of detections of the models Faster R-CNN and SSD.

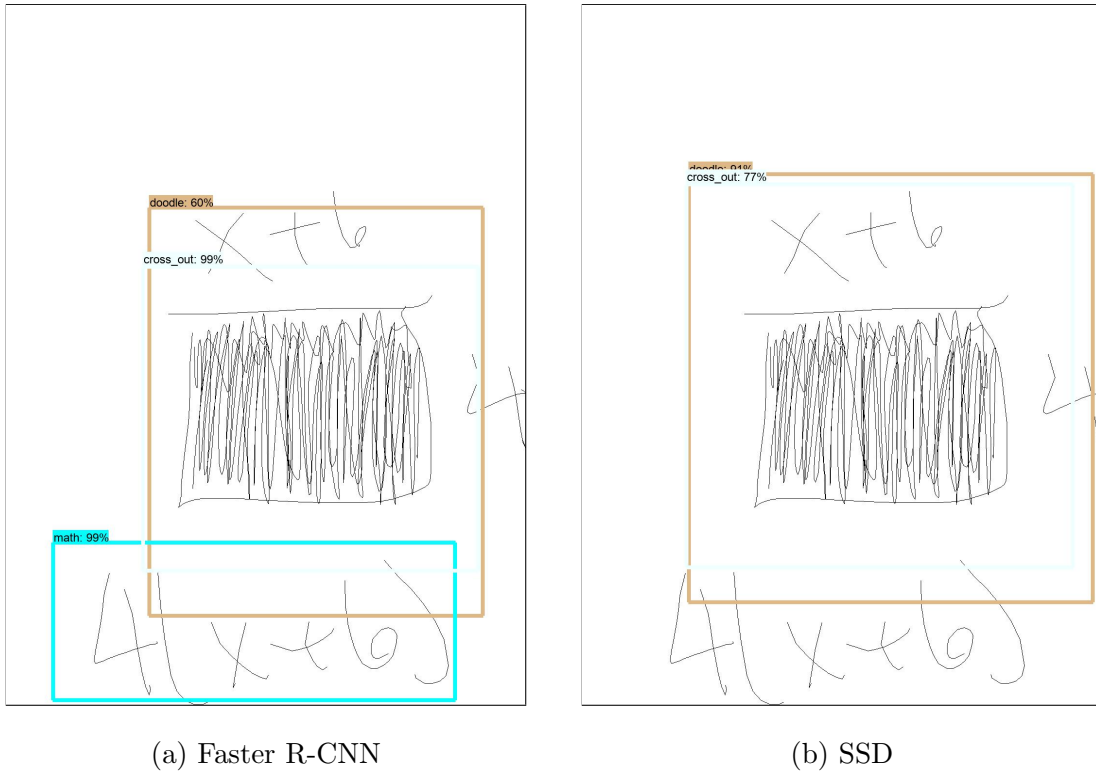
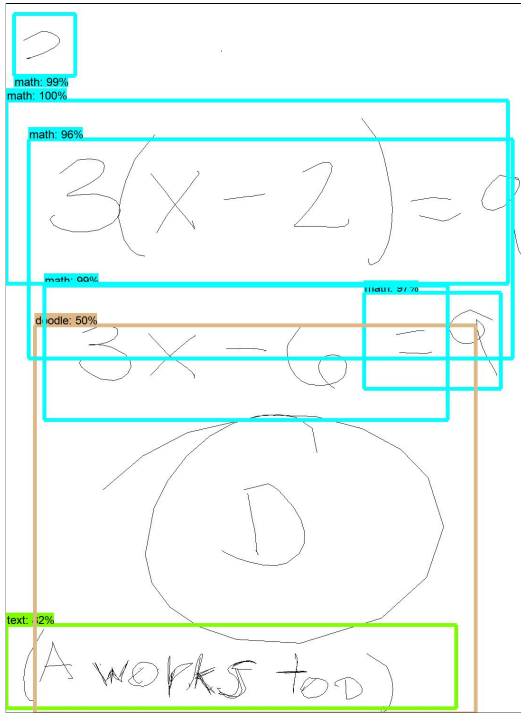
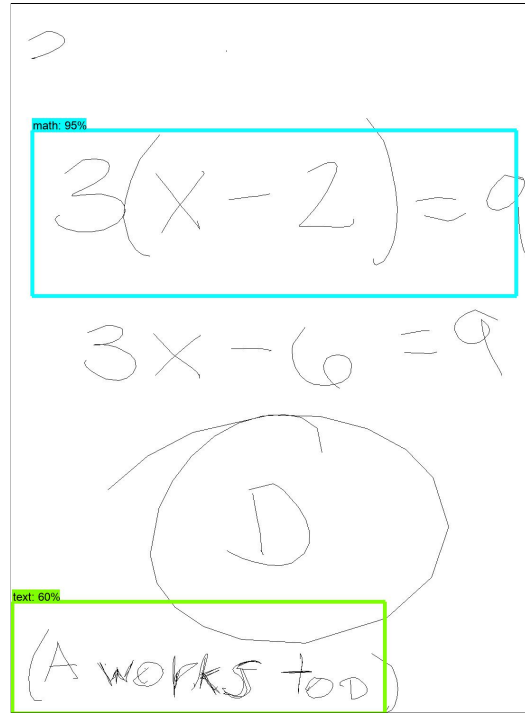


Figure 5.14: Side-by-side comparisons of detections of the models Faster R-CNN and SSD.

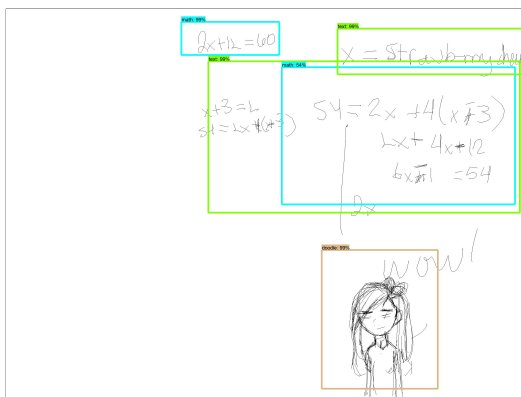


(a) Faster R-CNN

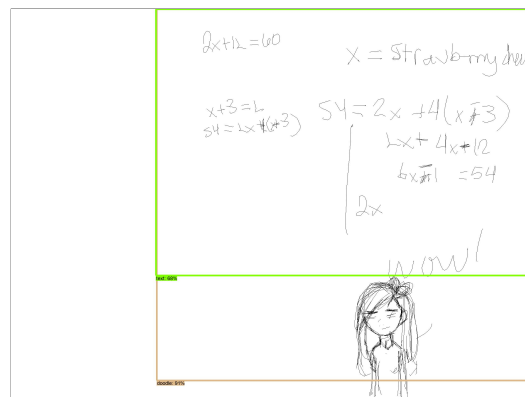


(b) SSD

Figure 5.15: Side-by-side comparisons of detections of the models Faster R-CNN and SSD.



(a) Faster R-CNN



(b) SSD

Figure 5.16: Side-by-side comparisons of detections of the models Faster R-CNN and SSD.

## Chapter 6

### POSSIBLE IMPROVEMENTS

In this section, we discuss possible improvements in the model and in the data. We also discuss how our data is best suited for the analysis of handwritten documents written by middle school students.

#### 6.1 Possible improvements in the model

The models Faster R-CNN and SSD were originally used to detect objects on natural images, which are colored (have 3 channels, RGB) and have noise in them. The handwritten data on the other hand is binary i.e. only black or white, has no noise and is only 1 channel. For the handwritten data, the model only has to learn the edges, shapes and their relations; it doesn't need to learn color dependencies with other objects and everything that relates to natural images. This leaves room for change in the model by reducing the number of layers in the network.

The data we collected can be represented with only 0's and 1's - 0 for white background and 1 for handwritten ink. This kind of reduction in data leaves room for a change in model to work faster.

#### 6.2 Possible improvements in the data

While gathering more data from students by conducting trials is a way of augmenting the handwritten corpus, another way is to transform the data by rotating (90°, 180°, 270°), flipping horizontally and vertically, and also by rearranging the ground truth at different positions. These data augmentation techniques not only increase the data but also are known to reduce over-fitting on image data (Krizhevsky

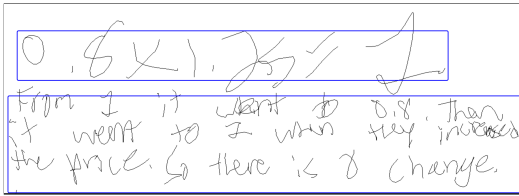


*et al.*, 2012)

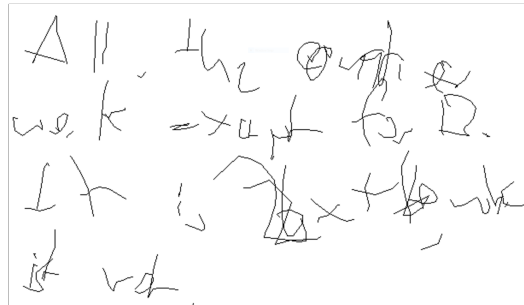
Another way of increasing the accuracy and decreasing the noise is to avoid the white background during prediction. Some of the predictions such as the one shown in Figure 5.2b has a prominent area of white space with detection as doodle. The empty space can be avoided to account for only handwritten data on the image. This will hopefully reduce the noise and increase the accuracy.

### 6.3 Complication with the data

Occasionally, the handwriting in the images were not legible as illustrated in Figures 6.1. The authors of this data are middle school students and thus our data is best suited for the analysis of handwritten document written by middle school students. The illegibility accounts for the noise and thus can be used as training if the test data is also from middle school students. However, if we were to work on a typical hand written document (Figure 6.2), compiled by university students or public in general, we can use the IAMonDo-database available at (Indermühle *et al.*, 2010b).



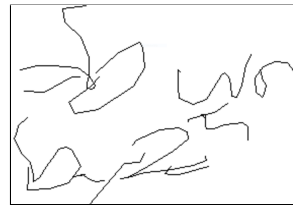
(a) Image with illegible handwriting.



(b) Image with illegible handwriting.



(c) Image with illegible handwriting.



(d) Image with illegible handwriting.

Figure 6.1: Sample images with illegible handwriting.

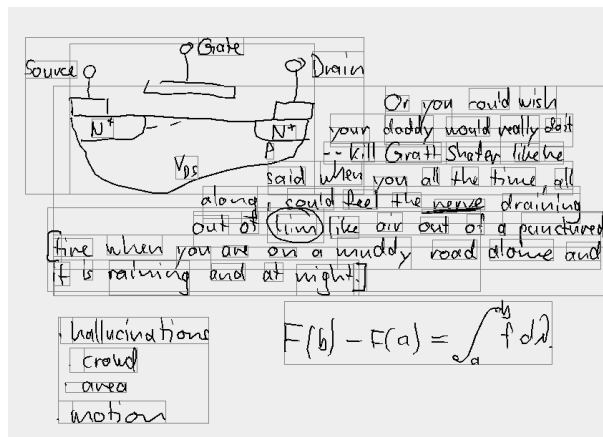


Figure 6.2: A sample handwritten document from the IAMonDo-database.

## Chapter 7

### CONCLUSION

We have gathered a handwritten image corpus, labelled it and used it to detect content type (text, math, diagram, cross out, table, graph, tick mark, arrow or doodle) using Faster R-CNN and SSD object detection techniques. We hope that this helps in further analysis tasks such as understanding a digital handwritten document.

## REFERENCES

- Bradley, A. P., “The use of the area under the roc curve in the evaluation of machine learning algorithms”, *Pattern recognition* **30**, 7, 1145–1159 (1997).
- Cheema, S. and J. LaViola, “Physicsbook: a sketch-based interface for animating physics diagrams”, in “Proceedings of the 2012 ACM international conference on Intelligent User Interfaces”, pp. 51–60 (ACM, 2012).
- Escalera, A. d. l., L. Moreno, M. A. Salichs and J. M. Armingol, “Road traffic sign detection and classification”, (1997).
- Everingham, M., S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn and A. Zisserman, “The pascal visual object classes challenge: A retrospective”, *International Journal of Computer Vision* **111**, 1, 98–136 (2015).
- Everingham, M., L. Van Gool, C. K. Williams, J. Winn and A. Zisserman, “The pascal visual object classes (voc) challenge”, *International journal of computer vision* **88**, 2, 303–338 (2010).
- Fu, L. and L. B. Kara, “From engineering diagrams to engineering models: Visual recognition and applications”, *Computer-Aided Design* **43**, 3, 278–292 (2011).
- Girshick, R., “Fast r-cnn”, arXiv preprint arXiv:1504.08083 (2015).
- Girshick, R., J. Donahue, T. Darrell and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation”, in “Proceedings of the IEEE conference on computer vision and pattern recognition”, pp. 580–587 (2014).
- Hanley, J. A. and B. J. McNeil, “The meaning and use of the area under a receiver operating characteristic (roc) curve.”, *Radiology* **143**, 1, 29–36 (1982).
- Hastie, T., R. Tibshirani and J. Friedman, “Unsupervised learning”, in “The elements of statistical learning”, pp. 485–585 (Springer, 2009).
- He, K., X. Zhang, S. Ren and J. Sun, “Deep residual learning for image recognition”, in “Proceedings of the IEEE conference on computer vision and pattern recognition”, pp. 770–778 (2016).
- Huang, J., V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama *et al.*, “Speed/accuracy trade-offs for modern convolutional object detectors”, in “IEEE CVPR”, (2017).
- Indermühle, E., H. Bunke, F. Shafait and T. Breuel, “Text versus non-text distinction in online handwritten documents”, in “Proceedings of the 2010 ACM Symposium on Applied Computing”, pp. 3–7 (ACM, 2010a).
- Indermühle, E., M. Liwicki and H. Bunke, “Iamondo-database: an online handwritten document database with non-uniform contents”, in “Proceedings of the 9th IAPR International Workshop on Document Analysis Systems”, pp. 97–104 (ACM, 2010b).

- Jain, K., A. M. Namboodiri and J. Subrahmonia, “Structure in on-line documents”, in “Document Analysis and Recognition, 2001. Proceedings. Sixth International Conference on”, pp. 844–848 (IEEE, 2001).
- Keyzers, D., F. Shafait and T. M. Breuel, “Document image zone classification—a simple high-performance approach”, in “in 2nd Int. Conf. on Computer Vision Theory and Applications”, (Citeseer, 2007).
- Kleinberg, J., C. Papadimitriou and P. Raghavan, “A microeconomic view of data mining”, *Data mining and knowledge discovery* **2**, 4, 311–324 (1998).
- Krizhevsky, A., I. Sutskever and G. E. Hinton, “Imagenet classification with deep convolutional neural networks”, in “Advances in neural information processing systems”, pp. 1097–1105 (2012).
- Lin, T.-Y., M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár and C. L. Zitnick, “Microsoft coco: Common objects in context”, in “European conference on computer vision”, pp. 740–755 (Springer, 2014).
- Liu, W., D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu and A. C. Berg, “Ssd: Single shot multibox detector”, in “European conference on computer vision”, pp. 21–37 (Springer, 2016).
- Nair, V. and G. E. Hinton, “Rectified linear units improve restricted boltzmann machines”, in “Proceedings of the 27th international conference on machine learning (ICML-10)”, pp. 807–814 (2010).
- Okun, O., D. Dörmann and M. Pietikainen, “Page segmentation and zone classification: the state of the art”, Tech. rep., OULU UNIV (FINLAND) DEPT OF ELECTRICAL ENGINEERING (1999).
- Onoro-Rubio, D. and R. J. López-Sastre, “Towards perspective-free object counting with deep learning”, in “European Conference on Computer Vision”, pp. 615–629 (Springer, 2016).
- Redmon, J., S. Divvala, R. Girshick and A. Farhadi, “You only look once: Unified, real-time object detection”, in “Proceedings of the IEEE conference on computer vision and pattern recognition”, pp. 779–788 (2016).
- Ren, S., K. He, R. Girshick and J. Sun, “Faster r-cnn: Towards real-time object detection with region proposal networks”, in “Advances in neural information processing systems”, pp. 91–99 (2015).
- Simonyan, K. and A. Zisserman, “Very deep convolutional networks for large-scale image recognition”, arXiv preprint arXiv:1409.1556 (2014).
- Stahovich, T. F. and H. Lin, “Enabling data mining of handwritten coursework”, *Computers & Graphics* **57**, 31–45 (2016).

- Strouthopoulos, C. and N. Papamarkos, “Text identification for document image analysis using a neural network”, *Image and Vision Computing* **16**, 12-13, 879–896 (1998).
- Szegedy, C., W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich *et al.*, “Going deeper with convolutions”, (Cvpr, 2015).
- VanLehn, K., S. Cheema, J. Wetzel and D. Pead, “Some less obvious features of classroom orchestration systems”, in “Educational Technologies: Challenges, Applications and Learning Outcomes”, (Nova Science Publishers, Inc., 2016).
- Wong, K. Y., R. G. Casey and F. M. Wahl, “Document analysis system”, *IBM journal of research and development* **26**, 6, 647–656 (1982).
- Yu, G. and J. Yuan, “Fast action proposals for human action detection and search”, in “Proceedings of the IEEE conference on computer vision and pattern recognition”, pp. 1302–1311 (2015).