A Study on Image Processing to Facilitate Business System by Multiple Barcode Detection

A thesis

Submitted in partial fulfillment of the requirements for the Degree of

Bachelor of Science in Computer Science and Engineering

Submitted by

Tasnim Mashrur Mahee	**150104013**
Atiqul Islam Chowdhury	**150104014**
Mushfika Sharmin Rahaman	**150104016**
Rifat Ahamad	**150104028**

Supervised by

Nazmus Sakib

Assistant Professor

Department of Computer Science and Engineering

Ahsanullah University of Science and Technology

Department of Computer Science and Engineering

Ahsanullah University of Science and Technology

Dhaka, Bangladesh

June, 2019

CANDIDATES' DECLARATION

We, hereby, declare that the thesis presented in this report is the outcome of the investigation performed by us under the supervision of Nazmus Sakib, Assistant Professor, Department of Computer Science and Engineering, Ahsanullah University of Science and Technology, Dhaka, Bangladesh. The work was spread over two final year courses, CSE4100: Project and Thesis-I and CSE4250: Project and Thesis-II, in accordance with the course curriculum of the Department for the Bachelor of Science in Computer Science and Engineering program.

It is also declared that neither this thesis nor any part thereof has been submitted anywhere else for the award of any degree, diploma or other qualifications.

Tasnim Mashrur Mahee
150104013

Atiqul Islam Chowdhury
150104014

Mushfika Sharmin Rahaman
150104016

Rifat Ahamad
150104028

CERTIFICATION

This thesis titled, **"A Study on Image Processing to Facilitate Business System by Multiple Barcode Detection"**, submitted by the group as mentioned below has been accepted as satisfactory in partial fulfillment of the requirements for the degree B.Sc. in Computer Science and Engineering in June, 2019.

Group Members:

Tasnim Mashrur Mahee	**150104013**
Atiqul Islam Chowdhury	**150104014**
Mushfika Sharmin Rahaman	**150104016**
Rifat Ahamad	**150104028**

Nazmus Sakib
Assistant Professor & Supervisor
Department of Computer Science and Engineering
Ahsanullah University of Science and Technology

Prof. Dr. Kazi A Kalpoma
Professor & Head
Department of Computer Science and Engineering
Ahsanullah University of Science and Technology

ACKNOWLEDGEMENT

First and foremost, We are grateful to the Almighty Allah for the good health and well being that were necessary to complete this thesis work. Then we have to thank our thesis supervisor, Nazmus Sakib - without whose encouragement, this thesis would have never been accomplished. His constant support, understanding and constructive critiques over the final year have enriched our thesis work to a great extent.

We place on record, our sincere thanks to Prof. Dr. Kazi A Kalpoma, honourable Head of the department, for her continuous encouragement.

We take this opportunity to express gratitude to all the respected faculty members of our department for their help and support. Also, we thank our parents for the unceasing encouragement, support and attention. Finally, we are grateful to everyone who helped us in every possible ways to make our thesis fruitful.

Dhaka
June, 2019

Tasnim Mashrur Mahee

Atiqul Islam Chowdhury

Mushfika Sharmin Rahaman

Rifat Ahamad

ABSTRACT

Barcoding system is a cheap and reliable way for tagging the products. Barcode detection process is needed for an invetory system to detect the barcodes of the products and for the billing system of the inventory concerned. Now a days, laser scanners are costly which is used to detect single barcode in supershops. If we can detect multiple barcodes from an image, it can help everyone to save some more time than scanning them separatelt. Previously there were many methods like filtering, feature extraction, YOLO model for detecting single barcodes in a fast and reliable way. All those methods did not focus to detect multiple barcodes accurately. In this thesis work, we have developed a model which works better for detecting and decoding multiple barcodes simultaneously. We have used Faster RCNN model for the detection of multiple barcodes. The detection process is also done using Pyzbar library separately. But Faster RCNN gives us better output to detect barcodes from an image than this library does. With the help of Tensorflow API, we worked on our dataset for the detection process of barcodes at first. After training the barcodes, the model is tested with more than three hundred images of high and low resolutions. The detection rate is much improved than the existing studies, for both resolutions and almost all the barcodes are effectively detected. Moreover, the decoding process is done on Arte-Lab dataset with the help of Zbar library. Though the detection rate using Faster RCNN is little bit slower, but it gives better accuracy. The detection and decoding accuracy throughout the model can facilitate business system for faster transaction.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

As consumers, we see barcodes and barcode scanners used all the time while purchasing from any retail store, renting a car, attending major events, flying, and even going to the doctor. They are in our social media apps and on store windows. A barcode is a machine-readable code in the form of numbers and a pattern of parallel lines of varying widths, printed on and identifying a product [2]. A bar code can best be described as an "optical Morse code." Series of black bars and white spaces of varying widths are printed on labels to uniquely identify items. The bar code labels are read with a scanner, which measures reflected light and interprets the code into numbers and letters that are passed on to a computer [2]. Because there are many ways to arrange these bars and spaces, numerous symbologies are possible. But in truth a barcode is so much more. Barcode systems help businesses and organizations track products, prices, and stock levels for centralized management in a computer software system allowing for improvement in productivity and efficiency.

1.1 Objective

Now we are living in such time where everything, every process in each sector are being automated. Advantages commonly attributed to automation include higher production rates and increased productivity, more efficient use of materials, better product quality, improved safety, shorter workweeks for labour and reduced factory lead times [1]. So, maximum profit will be possible surely if we properly use the automation process in economical sector. Automation requires no human interaction at all. But traditionally in super shops, we are used to see that a lot of human interactions are made. People buy products and then make one or more queue for scanning those products where some employees are appointed for scanning barcodes with a barcode scanner. It is a traditional view of a super shop. Here, we

have to focus on two things. Firstly, a lot of time is wasted by both consumers and sellers for the scanning method. Moreover, the barcode scanners which are normally used in super shop, are costly. So it is tough to start a new super shop or any business where barcodes are needed. For barcoding system, laser scanners are needed to detect and those scanners detect barcodes one by one which require more time. After analyzing these factors, we thought that if we use better version of barcode scanners to scan multiple products simultaneously, the problem would be solved easily. In this way we can easily minimize the time spent and can get rid of from long time spending at queue. That is how this idea can play a great role to maximize the productivity in a business.

1.2 Structure of a Bar

A barcode's density is determined by the "X" dimension. Density refers to the amount of information that can be captured in the bar code in a particular space, usually a linear inch. High density bar codes have low numbers and low density bar codes have high numbers. This is because individual characters consist of some combination of bars and spaces that are each multiples of "X". When "X" is small, the area required for each character is less than when "X" is large; thus the bar code can hold more per linear inch and is said to be of higher density [8]. Similarly, increasing the width of the narrowest element ("X") increases the space required for each character and reduces the number of characters per inch. Because the resulting code is often quite large, very low density codes are often associated with applications such as warehousing that require reading bar codes from a significant distance (3 to 30 feet).

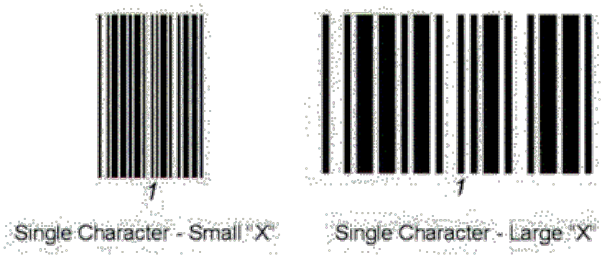

Figure 1.1: Barcode structure

All bar codes have start/stop characters that allow the bar code to be read from both left to right and right to left. Unique characters placed at both the beginning and end of each bar code, the stop/start characters provide timing references, symbology identification, and direction of read information to the scanner [8]. By convention, the unique character on the left of the bar code is considered the "start" and the character on the right of the bar code

is considered the "stop." Immediately preceding the start character and following the stop character is an area of no markings called the quiet zone. The quiet zone helps the scanner to find the leading edge of the barcode. As a rule, the quiet zone should be ten times the "X" dimension or 1/4, whichever is greater.

1.3 1D Barcode vs 2D Barcode

There are two types of barcodes which are mainly used now a days. They are: 1D barcodes and 2D barcodes. The 1D barcodes are linear barcodes which consist of vertical lines of varying widths with specific gaps resulting in a particular pattern. The 2D barcodes are more complex than 1D barcode. They encode data generally in square or rectangular patterns of two dimensions [9]. In general, 2D barcodes can represent more data per unit area. They support a bigger character set than 1D barcodes.

A linear barcode typically holds any type of text information. In contrast, a 2D barcode is more complex and can include more information in the code: price, quantity, web address or image. A linear barcode scanner cannot read a 2D barcode, requiring the use of an image scanner for reading the information embedded in a 2D barcode [9]. Mobile phones with cameras, like iPhones and Android phones, and many other devices can read 2D barcodes through their integrated cameras.

Two dimensional barcodes (2D) are also gaining popularity as more companies need larger storage capacities but have smaller places to put labels. Older "linear" bar codes resemble a picket fence (or ladder if they are positioned vertically) [10]. They use a single dimension - the width of the bars and spaces to read and decipher the code. The problem is that when we add more data to these lineal codes they get longer, eventually getting too big. Single dimension barcodes work well for labels with little content such as asset serial numbers or price tagging. Here at EIM, we often refer to a lineal barcode as a "license plate" which does not mean much until we use it to access your database where you keep a lot of other information.

1D barcodes can be scanned with traditional laser scanners, or using camera-based imaging scanners. 2D barcodes, on the other hand, can only be read using imagers. In addition to holding more information, 2D barcodes can be very small, which makes them useful for marking objects that would otherwise be impractical for 1D barcode labels. With laser etching and other permanent marking technologies, 2D barcodes have been used to track everything from delicate electronic printed circuit boards to surgical instruments [11].

1D barcodes, on the other hand, are well suited for identifying items that may be associated with other information that changes frequently. To continue with the UPC example, the item

the UPC identifies will not change, although the price of that item frequently does; that is why linking the static data (item number) to the dynamic data (the pricing database) is a better option than encoding price information in the barcode itself. 2D barcodes have been used in supply chain and manufacturing applications as the cost of imaging scanners has fallen. By switching to 2D barcodes, companies can encode more product data while making it easier to scan items as they move on assembly lines or conveyors and it can be done without worrying about scanner alignment.

This is especially true in the electronics, pharmaceutical, and medical equipment industries where companies have been tasked with providing a large amount of product tracking information on some very small items. For example, the U.S. FDA's UDI rules require several pieces of manufacturing information to be included on certain types of medical devices. That data could be easily encoded on very small 2D barcodes. While there is a difference between 1D and 2D barcode scanning, both types are useful, low-cost methods of encoding data and tracking items. The kind of barcode (or combination of barcodes) we select will depend on the specific requirements of your application, including the type and amount of data we need to encode, the size of the asset/item, and how and where the code will be scanned.

1.4 Extraction of Barcode

There are mainly two tasks for barcode readers: barcode locating and barcode decoding . Barcode localization methods have two objectives, speed and accuracy. For industrial environment, accuracy is crucial since undetected (missed) codes may lead to loss of profit. Processing speed is a secondary desired property of the detectors. On smartphones, the accuracy is not so critical, since the device interacts with the user and re-shoting is easily possible, but a fast (and reasonably accurate) barcode detection is desirable. Various techniques are used to locate and decode barcodes from photographs, from the classical line scanning technique [12], through the widely studied morphological approaches [13], and recent studies using wavelets [14]. Once an image is obtained, the first step is to localize the barcode, or to find its location within the image. Many methods to do this have been developed. Again, after the barcode is localized using one of the methods, it must be decoded in order to obtain the product's information. We have followed these processes by implementing this using Python. Firstly we have processed an algorithm to detect the barcode from the snap, then we have to decode it. And the main benefit in this process is, we can detect multiple barcodes at a time and can detect the barcode of cylindrical objects too. We have used Zamberletti algorithm using opencv and pyzbar library to detect and decode the barcodes from the snap.

While detecting the barcodes, there are some issues in Zambarletti algorithm. This algorithm is helpful to detect and decode barcodes simultaneously. But it does not detect properly, basically the blurry barcode or small regional barcode is not detected by this algorithm. In this case, Deep Learning approach has a better solution. Deep learning is a subset of machine learning where artificial neural networks, algorithms inspired by the human brain, learn from large amounts of data. Similarly, to how we learn from experience, the deep learning algorithm would perform a task repeatedly, each time tweaking it a little to improve the outcome. We refer to "Deep Learning" because the neural networks have various (deep) layers that enable learning. And RCNN is a method involved in Deep Learning. Based on the Region Proposal, RCNN fulfills the object detection using selective search for the feature extraction from CNN and SVM classification.

Chapter 2

Related Works

Barcode detection process were critical and less accurate in previous papers. The authors of some papers tried to implement the detection process using webcam, some of them focused on camera resolution, some of them proposed the detection process of 2D barcodes. Upasani et. al proposed a method of extracting information from the barcode at a lesser cost compared to typical electronic barcode scanners [15]. They had analyzed their diagram by following three levels, they are: level 0, level 1, level 2. They saved the product information in database. After that they take a snap with USB webcam and make ready the image for scanning by removing noise and eliminate unnecessary surrounding information. They perform cropping by observing the intensity of each pixel and extracting the rows of a barcode. By that time the contrast of the image improve to distinguish between bars by making the black bars one shade darker on the grayscale compared to white bars. Then the contrast-enhanced image is firstly binarized and edge detection is done successfully. After that the barcode number is decoded using the array of bar widths and then, after matching the product information, the product bill had been updated. EAN-13 barcode was used for this experiment.

Another work was about to detect single and plural barcodes which were detected at a time and the accuracy was 0.7 seconds to detect the barcodes [16]. The authors of the paper proposed a method which was about to extract candidate barcode skeletons to locate barcodes in an image. Their method was capable of searching rotated barcodes in a high-resolution image. Firstly the image was preprocessed by skeleton extraction in the downscaled image plane and verified barcode location in the original image plane. After the preprocessing, cross-scanning was employed to scan the binary downscaled image in a first direction, and then the binary downscaled image was scanned in a second direction perpendicular to the first direction. After that, a grading scheme was used to determine whether a detection window may contain portion of a barcode. The scheme was designed by exploiting the nature of barcodes as much as possible. It used the Sobel operator to track the main direction.

Tseng et. al proposed a method to detect QR codes from cylindrical object like bottles, cans [1]. These works influenced us to work on 1D and 2D barcodes at a time that means multiple barcode detection. Extraction of finder patterns was one of the major procedures for locating a QR code in a paper. In the preprocessing stage, source gray image was converted to binary image using the adaptive thresholding method [17] which is very robust and less computational complexity. Finder pattern candidates was located by roughly scanning and further confirmed by contour tracing [18] and corner detection. Three finder patterns located at the corners on the QR code had fixed ratio (1:1:3:1:1) of the widths of the black and white regions. While being scanned in arbitrary direction, these patterns were able to help quickly locating the QR code and speeding up the QR code reading. It obtained the candidates by roughly scanning in horizontal and vertical direction.

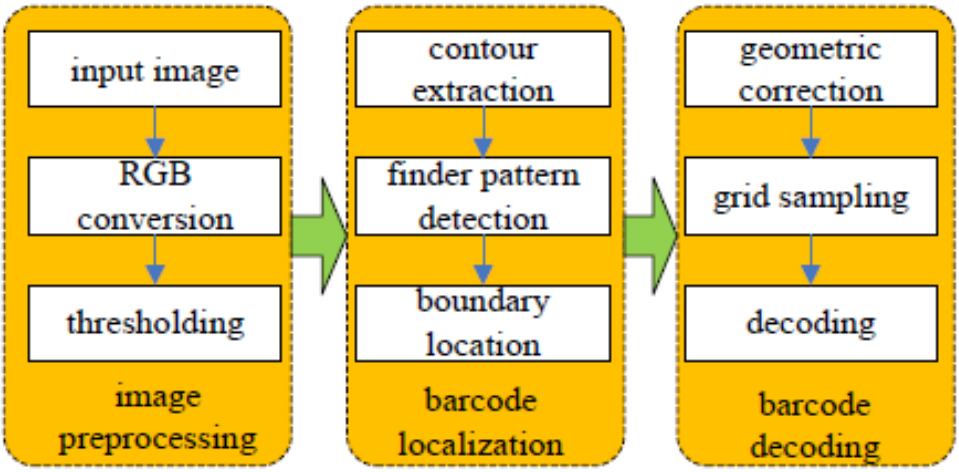

Figure 2.1: QR code recognition system overview [1]

After finding the three position detection patterns, a corner searching algorithm is run to obtain the four corner points as it is necessary to get the boundaries as a precondition to determine the fourth corner under the circumstances that other three corner points have been located previously. After obtaining boundary the curved QR code is transformed to the square plane through geometrical correction. Since QR codes might be perspective distorted because the camera can be held in a rotation or elevation angle.

Zhang et. al proposed a novle method on detecting barcodes through cross identification using mobile platform [19]. In this paper approach had been made for fast and robust color barcode detection. A feature detector was proposed for finding crosses formed by differently colored cells. Next, candidate barcode regions were generated using a thresholding method and those regions containing more than a specific number of crosses are declared to be barcode regions. Comparisons with the base line Hough transform method for matrix code detection were performed. To find the crosses, it used a cross detector to search for reliable X-shaped and T-shaped crosses. When the detector was placed at or near the center of a cross, each pixel in the probes records a color value. Then it combined different cross

detectors. After that, the cross detector might still get false positive responses in identifying cross feature points. For this reason false crosses were filtered where the standard deviation of pixel values is fairly small. After performing its detector on the input image, candidate barcode regions were generated by Niblack's thresholding method, which was robust to varying light conditions and complex background. Finally, shape consistency tests were performed on detected barcode regions to further exclude false detections.

Lin et. al proposed a method of identifying the QR and Aztec barcodes by using connected component labeling algorithm to compute the tag for each connected component, then search for the innermost connected component of the finder patterns in the tag image [20]. The method could identify rotated or defocused barcode images. At first it labeled the connected components in the barcode image. Standard thresholding algorithms, such as the Otsu's algorithm, were used for thresholding [21] . The CCL algorithm assigned every connected component a unique number, called the tag. This algorithm scanned the image from left to right and from top to bottom. Since the tag value increased in the scan direction, it was understandable that the innermost component in the finder pattern was a local maximum in the tag image.

Another work has been done on making this system angle invariant and less or not at all user interactive [22]. Nowadays mobile applications has been using to identify products from pictures and also that can make online comparisons with the existing similar products. This approaches mainly focus on decoding degraded barcodes and treat the underlying barcode detection task as a side problem that can be addressed using appropriate object detection methods. However, the majority of modern mobile devices do not meet the minimum working requirements of complex general purpose object detection algorithms and most of the efficient specifically designed barcode detection algorithms require user interaction to work properly. But in this work, they presented a novel method for barcode detection in camera captured images based on a supervised machine learning algorithm. The work identified one dimensional barcodes in the two-dimensional Hough Transform space and remarkably it was angle invariant, requires no user interaction and can be executed on a modern mobile device. Firstly, in this process, the rotation angle of the barcode was determined with Hough space using Multi Layer Perceptron (MLP). Galamhos et. al determined the set of all the segments of the image by applying the same technique of Matasyz [23]. After that, two histograms were defined that describe the intensity profile of the rows and the columns. Finally, a smoothing filter was applied to each histogram to remove low value bins corresponding to isolated non-barcode segments.

Some people works on the segmentation of images with 1D barcode, but also analyze the operation of different methods for 2D barcode images as well [24]. They tried to detect automatically, rapidly and accurately the barcode location by the help of extracted features. A novel algorithm was proposed that outperforms the other literatures in both accuracy and

efficiency in detecting 1D codes. The algorithm was based on based on bottom-hat filtering and other morphological operations.

Another work proposed that it might be used to detect and decode multiple 2D barcodes (i.e. Datamatrix) [25]. Most of the techniques discussed in literature, work for single 2D barcode detection and decoding. The barcodes were not wrapped in any reflective material such as polyethylene foiling. The proposed work however, not only used image processing techniques but also proposed an optical hardware setup to address the issues related to decoding of 2D barcodes under reflection and poor illumination conditions. The captured image contained multiple 2D barcodes and each barcode was located on a white rectangular region and first, they located the white rectangular patterns. Taking each white pattern as the search region, a 2D barcode was detected and decoded within this area using Cog2DSymbol tool. The decoded 2D barcodes could then be stored or transferred to the other devices over a network using Ethernet/IP.

There was a paper which was about to detect barcodes using web-cam. The paper proposed an angle invariant method for barcode detection in camera captured images based on the properties of the Hough Transform [15]. A properly trained supervised machine learning model identifies the rotation angle of every barcode in real world images by analyzing their Hough Transform spaces and a subsequent phase detected the bounding boxes surrounding those barcodes. They proved that their method can obtain excellent results for three different 1D barcode datasets and that it was also effective in detecting barcodes that were twisted, partially occluded or illegible due to reflections.

There was another paper which implements the barcode detection process using deep learning. They mainly followed YOLO method for the detection process. They described how to adapt the state-of-the-art deep learning-based detector of You Only Look Once (YOLO) for the purpose of detecting barcodes in a fast and reliable way. The detector is capable of detecting both 1D and QR barcodes [26]. The work was about to locate the barcode region from the product.

After analyzing those works, we have planned to implement a method which will accurately detect the barcodes in any resolution. And the decoding process will also help to store the product information. Because there were some limitations like these in previous papers and it is a challenging issue for us to operate these two process simultaneously.

Chapter 3

Background Study

After analyzing the related works on barcode detection, we have found that most of the works are about to detect single 1D barcode and multiple (maximum 3 or 4) barcode. But those works need much time for the detection process and they have slower accuracy rate. While exploring previous works, we have found some terms which are explained in the following section.

3.1 Barcode structure

A barcode consists of following points:

- **Margin:** Spaces, normally white, where nothing printed are required at each end of the code and they should be 10 times that of a narrow bar.
- **Barcode symbol:** The area composed of bars and spaces is known as the "bar code symbol".
- **Barcode:** The symbol together with the left and right margins make up what is called bar code.
- **Start character:** Indicates the start of the data. Depending on code system, start character varies.
- **Data(message):** Area indicating the actual data.
- **Check digit:** Confirms that there are no error in reading by means of one digit of compiled data.
- **Stop character:** Indicates the end of the data.

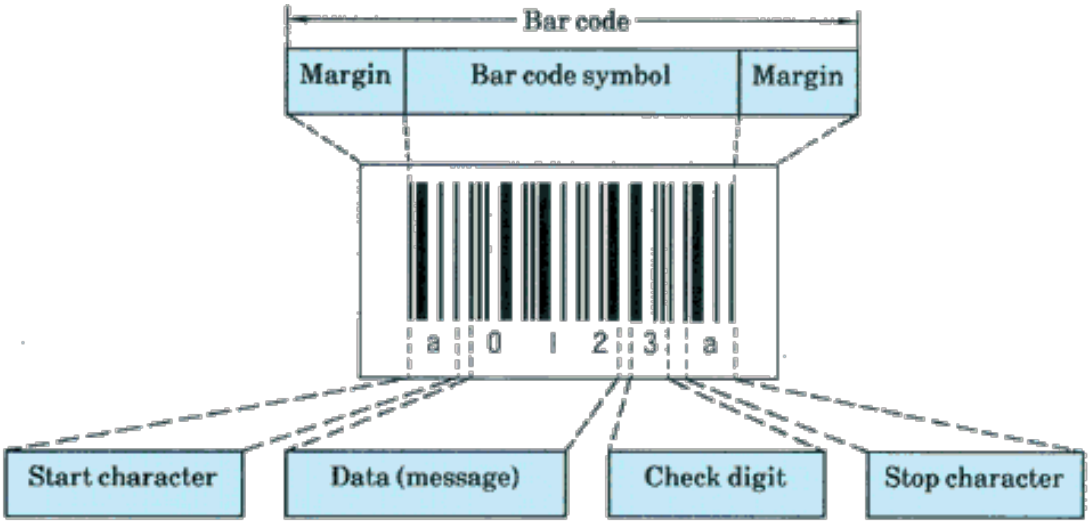

Figure 3.1: Structure of a 1D barcode [2]

3.2 Types Of Barcode

There are two types of barcodes: linear or 1D, and 2D. The most visually recognizable, the UPC (Universal Product Code) is a linear 1D barcode made up of two parts: the barcode and the 12-digit UPC number. The first six numbers of the barcode is the manufacturer's identification number. The next five digits represent the item's number. The last number is called a check digit which enables the scanner to determine if the barcode was scanned correctly.

3.2.1 1D Barcode font

3.2.1.1 Uniform Product Code (UPC)

"UPC" stands for Universal Product Code. UPC bar codes were originally created to help grocery stores speed up the checkout process and keep better track of inventory, but the system quickly spread to all other retail products because it was so successful. UPC (technically refers to UPC-A) consists of 12 numeric digits, that are uniquely assigned to each trade item. Along with the related EAN barcode, the UPC is the barcode mainly used for scanning of trade items at the point of sale.

Figure 3.2: Uniform Product Code (UPC) barcode

3.2.1.2 Code 39 (Code 3 of 9)

The Code 39 specification defines 43 characters, consisting of digits 0-9, the letters A-Z (upper case only), and the following symbols: space, minus (-), plus (+), period (.), dollar sign, slash (/), and percent. A special start/stop character is placed at the beginning and end of each barcode. The barcode may be of any length, although more than 25 characters really begins to push the bounds of practical physical width. An additional character is used for both start and stop delimiters. Each character is composed of nine elements: five bars and four spaces. Three of the nine elements in each character are wide (binary value 1), and six elements are narrow (binary value 0). The width ratio between narrow and wide is not critical, and may be chosen between 1:2 and 1:3.

Figure 3.3: Code 39 (Code 3 of 9) barcode

3.2.1.3 PostNet

The PostNet barcode is used by the United States Postal Service to automatically sort mail. The PostNet code consists of evenly spaced bars of two different heights. Each character is represented by five bars, two tall and three short. The character set includes the digits 0 through 9. The code begins and ends with a tall bar, and may contain a 5-digit ZIP code, a 9-digit ZIP+4 code, or an 11-digit Delivery Point Code. A Modulo 10 check digit ('correction character') is inserted after the ZIP code and before the ending frame bar.

Figure 3.4: PostNet barcode

3.2.1.4 Code 128

Code 128 is a high-density linear barcode symbology defined in ISO/IEC 15417:2007. It is used for alphanumeric or numeric-only barcodes. It can encode all 128 characters of ASCII.

Figure 3.5: Code 128 barcode

3.2.1.5 Bookland

The Bookland EAN barcode is used internationally to identify books as well as video and audio cassettes and software. The unique number assigned to each item is the International Standard Book Number (ISBN).

Figure 3.6: Bookland barcode

3.2.1.6 Interleaved 2 of 5

Interleaved 2 of 5 (ITF) is a numeric only barcode used to encode pairs of numbers into a self-checking, high-density barcode format. In this symbology, every two digits are interleaved with each other to create a single symbol. If a number string containing an odd number of digits needs to be encoded, a leading zero must be added to produce an even number of digits in the Interleaved 2 of 5 barcode.

Figure 3.7: Interleaved 2 of 5 barcode

3.2.1.7 Codabar

Codabar is the barcode developed by Monarch Marking Systems in 1972. It is the barcode introduced at early stage following "2 of 5". It is widely used for applications that require serial numbers, such as management of blood banks, slips for door-to door delivery services and member cards. Codabar has 4 bars and 3 spaces (total 7 elements) with each narrow or wide width representing one character (letter). Codabar can represent characters including numbers (0 to 9), letters (A, B, C, D) and symbols (space, minus, plus, period, dollar sign, slash).

Figure 3.8: Codabar barcode

3.2.2 2D Barcode font

3.2.2.1 PDF417

Large amounts of text and data can be stored securely and inexpensively when using the PDF417 barcode symbology. The printed symbol consists of several linear rows of stacked codewords. Each codeword represents 1 of 929 possible values from one of three different clusters. A different cluster is chosen for each row, repeating after every three rows. Because the codewords in each cluster are unique, the scanner is able to determine what line each cluster is from.

Figure 3.9: PDF417 barcode

3.2.2.2 Data Matrix

Data Matrix is a very efficient, two-dimensional (2D) barcode symbology that uses a small area of square modules with a unique perimeter pattern, which helps the barcode scanner determine cell locations and decode the symbol. Characters, numbers, text and actual bytes of data may be encoded, including Unicode characters and photos.

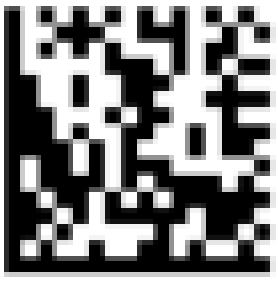

Figure 3.10: Data Matrix

3.2.2.3 Maxicode

Maxicode is an international 2D (two-dimensional) barcode that is currently used by UPS on shipping labels for world-wide addressing and package sortation. MaxiCode symbols are fixed in size and are made up of offset rows of hexagonal modules arranged around a unique finder pattern. MaxiCode includes error correction, which enables the symbol to be decoded when it is slightly damaged.

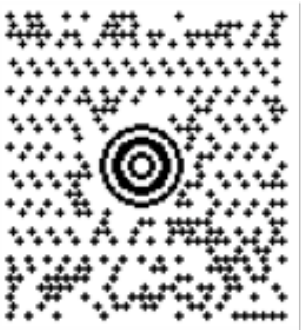

Figure 3.11: Maxicode

3.2.2.4 QR Code

QR-Code is a two-dimensional (2D) barcode type similar to Data Matrix or Aztec, which is capable of encoding large amounts of data. QR means Quick Response, as the inventor intended the symbol to be quickly decoded. The data encoded in a QR-Code may include alphabetic characters, text, numbers, double characters and URLs. The symbology uses a small area of square modules with a unique perimeter pattern, which helps the barcode scanner determine cell locations to decode the symbol.

Figure 3.12: QR Code

For our detection process, we have used Faster RCNN model which gives us better accuracy, but it is a little bit slower. The following section will discuss about the model.

3.3 Faster R-CNN

For the main steps of our methodology, we had to gain knowledge on Deep Learning. And, RCNN is mainly a deep learning approach. However, RCNN request a large memory from the disk for the pre-extraction of multiple region proposal. And Faster RCNN is as like as RCNN which accelerates the detection process. Faster RCNN is an object detection architecture presented by Ross Girshick, Shaoqing Ren, Kaiming He and Jian Sun in 2015, and is one of the famous object detection architectures that uses convolution neural networks like YOLO (You Look Only Once) and SSD (Single Shot Detector). Faster R-CNN has three parts: Convolution layers, Region Proposal Network (RPN) and Classes and Bounding Boxes prediction [26]. Fast R-CNN passes the entire image to ConvNet which generates regions of interest. Also, it uses a single model which extracts features from the regions, classifies them into different classes, and returns the bounding boxes. Faster R-CNN is good for detection than any other classification algorithm, because it gives more accuracy than any other classifier. It separates an image into different regions or bounding boxes, and train the image on its own way.

The Faster R-CNN works by using some steps, which are:

- Run the image through a CNN to get a Feature Map
- Run the Activation Map through a separate network, called the Region Proposal Network (RPN), that outputs interesting boxes/regions
- For the interesting boxes/regions from RPN use several fully connected layer to output class + Bounding Box coordinates

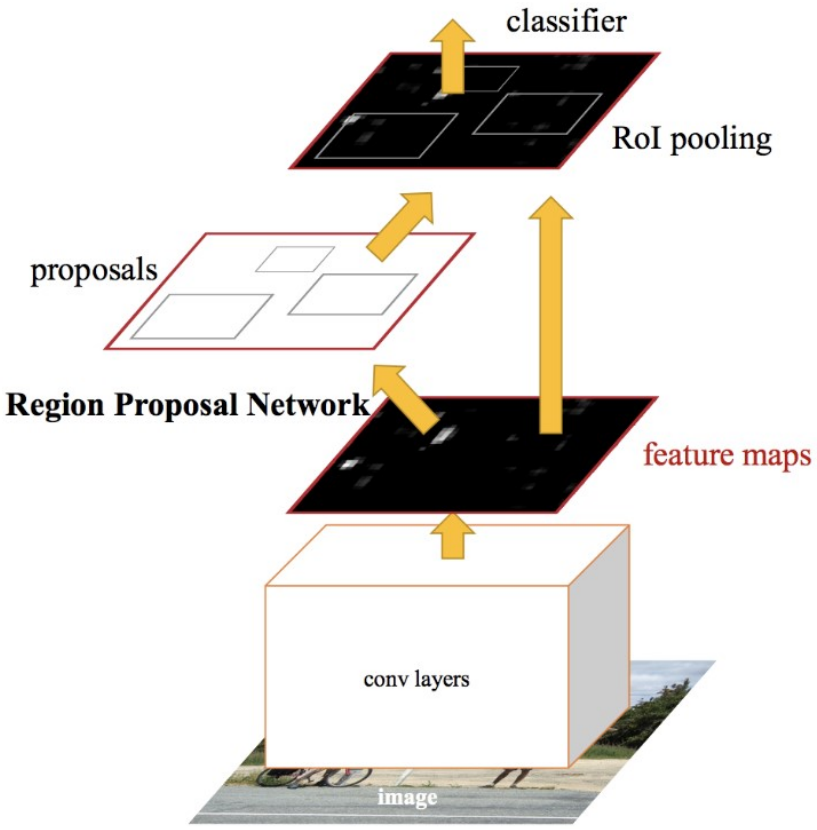

Figure 3.13: Faster R-CNNN Model [3]

Faster-RCNN is composed of 3 neural networks: Feature Network, Region Proposal Network (RPN), Detection Network. The Feature Network is usually a well-known pre-trained image classification network such as VGG minus a few last/top layers. The function of this network is to generate good features from the images. The output of this network maintains the shape and structure of the original image. The RPN is usually a simple network with three convolutional layers. There is one common layer which feeds into two layers - one for classification and the other for bounding box regression. The purpose of RPN is to generate a number of bounding boxes called Region of Interests (ROIs) that has high probability of containing any object [27]. The output from this network is a number of bounding boxes identified by the pixel co-ordinates of two diagonal corners, and a value. The Detection

Network takes input from both the Feature Network and RPN, and generates the final class and bounding box. It is normally composed of 4 Fully Connected or Dense layers. There are 2 stacked common layers shared by a classification layer and a bounding box regression layer. To help it classify only the inside of the bounding boxes, the features are cropped according to the bounding boxes. The main complexity here is the RPN and Detection Network need to be trained.

3.3.1 Feature Map

The feature map is the output of one filter applied to the previous layer. A given filter is drawn across the entire previous layer, moved one pixel at a time. Each position results in an activation of the neuron and the output is collected in the feature map. You can see that if the receptive field is moved one pixel from activation to activation, then the field will overlap with the previous activation by (field width - 1) input values.

Convolutional Neural Networks look for "features" such as straight lines [28]. As such whenever you spot those features-they get reported to the feature map. Each feature map is looking for something else. One feature map could be looking for straight lines, the other for curves. The feature maps also look for their features in different locations.

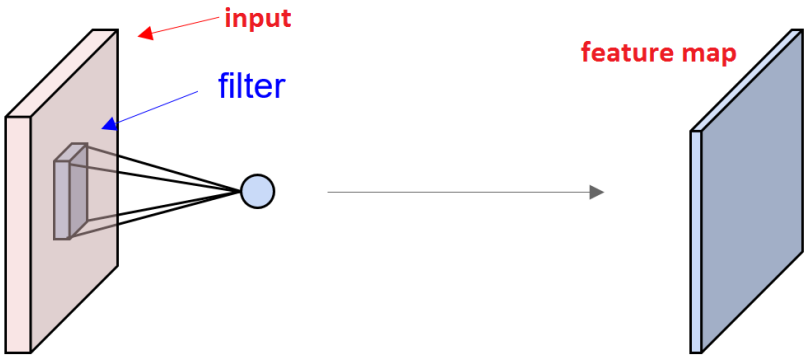

Figure 3.14: Feature Map Model [4]

3.3.2 Region Proposal Network (RPN)

In object detection using RCNN, RPN is the one true backbone and have proven to be very efficient till now. It's purpose is to propose multiple objects that are identifiable within a particular image.

Many people implement Faster R-CNN to identify the objects but this algorithm specifically dwells into the logic and math behind how algorithm gets the box around the identified objects. The developers of algorithm called it Region Proposal Networks abbreviated as

RPN. To generate these so called "proposals" for the region where the object lies, a small network is slide over a convolutional feature map that is the output by the last convolutional layer.

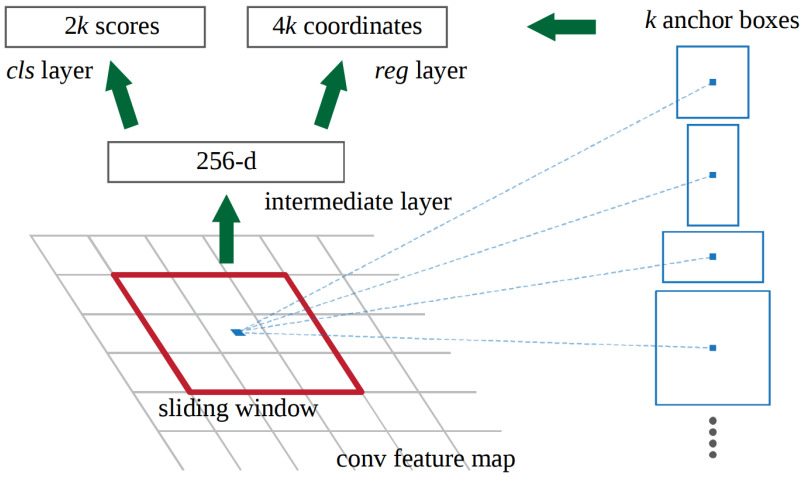

Figure 3.15: Region Proposal Network [5]

3.3.3 VGG Network Architecture

A Convolutional Neural Network (CNN, or ConvNet) are a special kind of multi-layer neural networks, designed to recognize visual patterns directly from pixel images with minimal preprocessing. VGGNet consists of 16 convolutional layers and is very appealing because of its very uniform architecture.

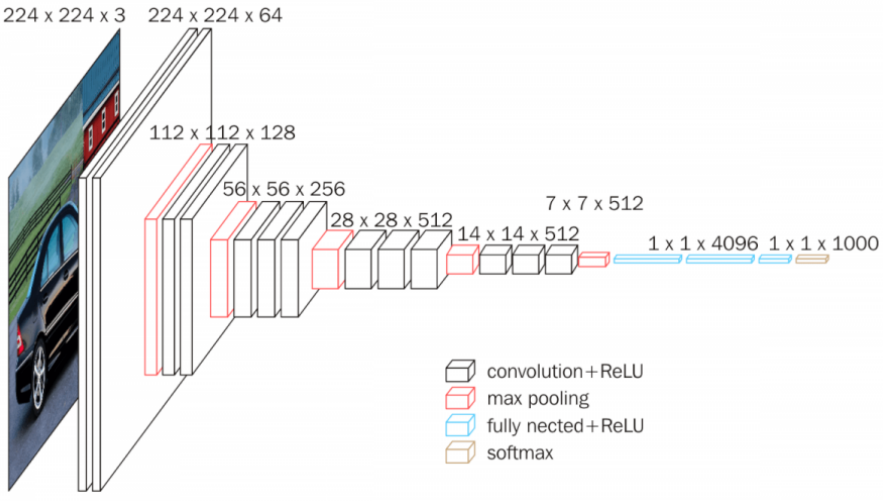

Figure 3.16: VGG16 Network Architecture [6]

VGGNet is similar to AlexNet, only 3x3 convolutions, but lots of filters. It is currently the most preferred choice in the community for extracting features from images. The weight

configuration of the VGGNet is publicly available and has been used in many other applications and challenges as a baseline feature extractor. The most used model is VGG16, which is used in Faster R-CNN. The main contribution of VGG is to show that classification/localisation accuracy can be improved by increasing the depth of CNN inspite of using small receptive fields in the layers. (especially earlier layers). Neural networks prior to VGG used bigger receptive fields (7*7 and 11*11) as compared to 3*3 in VGG, but they were not as deep as VGG. There are few variants of VGG, the deepest one is with 19 weight layers.

Chapter 4

Proposal

4.1 Motivation

Barcoding system is very important for business system. To detect barcodes from products, single laser scanners are used everywhere. And these laser scanners are costly. Again scanning barcodes one by one needs much more time. So our approach is to detect multiple barcodes using mobile camera sensor and decode them for the products information. We think that if multiple barcodes are detected simultaneously, then it will be less time consuming as well as less costlier. Our idea will help specially those sectors where multiple barcode detection and decoding process would make a system faster than before ever, thus increasing the productivity.

4.2 Proposed Method

Deep learning surrounds us every day and this will only increase with time. Whether you are thinking about cars that drive autonomously or even have some new technology like parking assistance, traffic control, or face recognition technology. Reports on events are likewise written by computers, more precisely: Computers using deep learning technologies to remain a constant learning process. This learning process enables the communication between computers and humans. Not only their function will increase over time, it will impact most industries as well as the jobs that come with them. Eventually it will impact all of us possibly on a daily basis. Corporate learning is so important. The era of digitalization has allowed the technology to flourish, and computers that are able to analyze massive amounts of complex data which can now provide more accurate results.

Earlier barcode decoding and detection processes were all about image processing. But

now a days, deep learning processes are helpful in this area. Already some paper has been published based on deep learning on this purpose but they worked on YOLO model and CNN model. They have already outperformed the previous techniques to detect and decode barcodes. So, we would like to investigate whether the use of deep learning can benefit the locating of barcodes by using Faster RCNN model.

Barcode detection system is very important for all kinds of shop, so that the products of the shop can be arranged in a better way. After reviewing all of these papers, we thought that there were many scopes to improve the method for barcode detection in many ways. We worked on multiple 1D barcode detection. We can also search for better filtering option without degrading the accuracy. We would put effort to detect barcode on complex background.

For this project, our initial plan was to work on some portions. We have got success on those task. The working portions are given below:

- Single 1D and 2D barcode detection and decoding
- Multiple 1D and 2D barcode detection and decoding
- Angle invariant barcode detection and decoding
- Barcode detection in complex background with higher accuracy

There are many types of barcode. 1D and 2D barcode are the main two types of barcodes. Single 1D barcode detection process are done with the help of Faster RCNN. For multiple barcode detection, we have to crop the portion of those 1D barcodes one by one. The detection process thus works well. Again, for the decoding process we have used OpenCV library with Spyder IDE. The type and value will be shown after the decode operation.

Our method is to detect multiple 1D and 2D barcodes from a snap. We have used EAN-13, Code 128 as 1D barcode and QR Code as 2D barcode. For the detection of the barcodes, we have followed two steps. Those two steps are:

- To find the position of a barcode
- To decode the barcode

This method has improved the detection of 1D and 2D barcodes at a time which was not easy earlier. Figure 4.1 shows the flow diagram of our working process. Our algorithm works smoothly with the help of some python library.

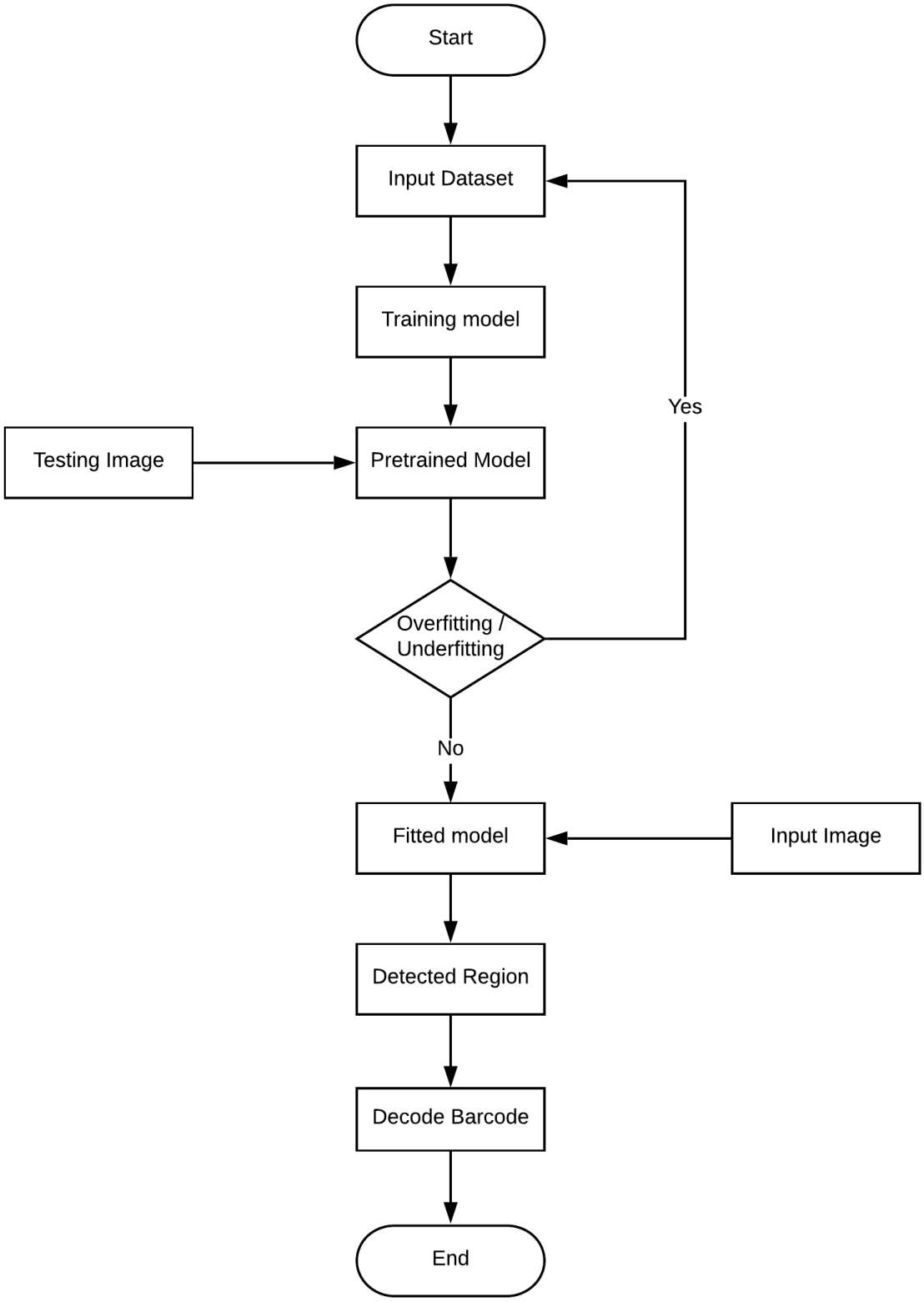

Figure 4.1: Flowchart of Barcode Detection Method

4.2.1 Barcode Detection

Barcode detection is required in a wide range of real life applications. Computer vision algorithms vary considerably and each application has its own requirements. For our method, pyzbar library was used for this purpose which implements raw image processing techniques at first. The result was very good in terms of single barcode detection. In some cases such as dark lighting condition, reflection, occulted barcodes it was providing very good result as well as in terms of multiple barcode detection it was often failed to distinguish the boundary regions. For better result deep learning idea was implemented.

A deep neural network provides state-of-the-art accuracy in many tasks, in object detection paradigm. The idea was to treat each barcode itself as an object rather than the product. Deep learning gives better accuracy because it learns automatically, without predefined knowledge explicitly coded by the programmers [29]. A neural network as the name suggests was inspired by biological neural system. The complexity of the human neural system was the vision of the software based neural network. The neural network in a person's brain is a hugely interconnected network of neurons, where the output of any given neuron may be the input to thousands of other neurons [30]. Like biological neural system deep neural network consists of layers. Each layer represents a deeper level of knowledge, i.e., the hierarchy of knowledge. More layers gives complex understanding of the system.

Faster R-CNN family of R-CNN was applied to find desired output. There are many models such as CNN, R-CNN, Fast R-CNN is in place. These models are mostly used in unstructured data e.g., image in order to extract information from it.

Convolution Neural Network (CNN) is used for image classification. A typical CNN can only predict the class of the objects but not where they are located. While an R-CNN, with the R standing for region, is for object detection. To bypass the problem of selecting a huge number of regions, instead of trying to classify a huge number of regions, proposed method just works with 2000 regions. The regions in the R-CNN are detected by selective search algorithm followed by resizing so that the regions are of equal size before they are fed to a CNN for classification and bounding box regression [30].

Faster R-CNN, the image is provided as an input to a convolutional network which provides a convolutional feature map. Instead of using selective search algorithm on the feature map to identify the region proposals, a separate network is used to predict the region proposals. The predicted region proposals are then reshaped using a ROI pooling layer which is then used to classify the image within the proposed region and predict the offset values for the bounding boxes [31].

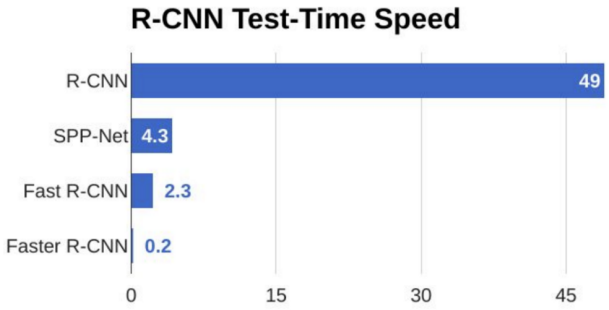

Figure 4.2: Comparison of test-time speed of object detection algorithms [7]

4.2.2 Barcode Decoding

Barcode detection process is needed to count the product barcodes from a snap. But to store the product information, billing information in a super shop, barcode decoding is necessary. The "type" and "number" of a barcode are needed to store the information. For the decoding process, we take help of Pyzbar library. The image is converted in Zbar compatible format. While decoding the barcode, the type, data and location information need to be extracted. Moreover, the decoding process works on single barcodes. The detected portion of barcode is cropped one by one and then decoded by using the Pyzbar library which stroes the data of the barcodes.

Chapter 5

Implementation

5.1 Overview

As previous studies have proven that the Zamberletti algorithm proved most efficient in the case of detection and decoding barcodes from an image [22]. Zamberletti algorithm was built on a deep learning model, this pre-trained model is not available publicly. This algorithm worked very well while detecting and decoding single different types of barcodes. Multilayer feedforward neural network based on the backpropagation algorithm for image restoration. We focused on extending and leveraging the idea of using deep learning from this algorithm, for multiple barcode detection - which is the focus of our work.

Zbar library is most commonly used to read a single barcode from images, the library provides detection and decoding steps. Some factors were noticed while using Zbar library because the detection depends on image quality and image enhancing issue. For the detection of the barcodes, the filtering and thresholding process is done by ZBar library as it is mainly designed for detecting the bars and portions of the barcodes. The detection process depends on the focus of the camera sensor. There is the minimum focal length needed to capture a better image so that the barcodes from the image can easily be detected. The other important factor is the resolution, which is described earlier. The quality of an image depends on the resolution of the image and so the barcodes can easily be detected by it. Image enhancing, the image quality of these factors role should be minimized while detecting the barcodes. While detecting multiple barcodes some other factors plays a significant role like camera position and object position while taking the image. Background on which object is kept and the shape of the objects.

Our goal was to find an exact solution like Zamberletti algorithm for detecting and decoding multiple barcodes. Our proposed idea was divided into two processes that is detection and decoding. Firstly, we thought of conquering an accurate detection process then the decoding

process. Our idea is that computers can analyze multiple barcodes parallelly. For computers be able to analyze barcodes the first task is to identify barcodes, that can be achieved using a deep learning based detector.

We thought of using deep learning with object detection purposes. Object Detection is the process of finding real world object instances like car, bike, TV, flowers, and humans in still images or videos. It allows for the recognition, localization, and detection of multiple objects within an image which provides us with a much better understanding of an image as a whole. Deep learning is an AI function which is computers imitating the working of the human brain processing data, recognizing patterns to use in decision making. Barcode is merely a pattern whenever the word barcode is used it creates an image of black and white stripes on the human mind. With the increased use of cognitive technologies, computers have become very smart in identifying patterns. The initial task to be completed is to tutor computers to locate the barcode regions from an image. Once the barcodes are detected then the detected regions could be separated. The decoding process could be applied to these separated images by the Zbar library. Zbar library has shown satisfactory results decoding single barcodes.

5.2 Platforms used for Experiment

Many platforms have been used for the detection of the barcodes. We are making a deep learning-based detector. Deep learning has various architectures. It is known that deep learning is a part of a family of machine learning methods which is based on artificial neural networks. The deep learning architecture we used is a convolution neural network. Faster RCNN is used for building our detector. The detector was trained on the TensorFlow object detector. This object detectors' usefulness is described in the following section.

We mainly used Python language to run the experiment. Python is an open source programming language that was made to be easy-to-read and powerful language. Python is not only an easy-to-use language but also is enriched with different types of libraries. It is stated before TensorFlow API is used for building the detector, the API is built on python.

The detector was initially trained on a GPU. The model of the GPU is gtx 1050. After completing the training, the detector works on any version of windows both CPU and GPU.

Another important part is the use of OpenCV library, which is necessary for our process. OpenCV-python is a library of Python bindings designed to solve computer vision problems. OpenCV does not have any dedicated modules that can be used to read and decode barcodes. However, OpenCV can facilitate the process of reading barcodes, including loading an image, grabbing a new frame from an image, and processing it. Once the image or frame is loaded

we can then pass it to a dedicated Python barcode decoding library such as a Zbar. The ZBar library will then decode the barcode or QR code. OpenCV can come back in to perform any further processing and display the result.

ZBar barcode reader library helps us to improve the detection process of the barcodes. It is an open source software suitable for reading bar codes from various sources, such as video streams, image files, and raw intensity sensors. It supports EAN-13/UPC-A, UPC-E, EAN-8, Code 128, Code 39, Codabar, Interleaved 2 of 5 and QR Code.

5.3 Detection of Barcode

As stated earlier, we are using deep learning with object detection purposes. Our idea was to treat the barcodes as objects. We failed to find any pre-trained model serving our purpose. Deep learning has various architectures. Faster RCNN is one of the most used object detection neural network. We have made just a single image classifier to train the model. For training in custom dataset TensorFlows' object detection API is used. TensorFlows' Object Detection API is an open source framework built on top of TensorFlow that makes it easy to construct, train and deploy object detection models.

TensorFlow object detector provides us with models that were trained on the COCO (common object contexts) dataset and works well on the 90 commonly found objects included in this dataset. The beauty of the API is that it can train on a new object that is not part of the COCO. Our training was done single image classifier that is 1D barcodes. Primarily for experimenting with our idea, we trained our model with few images it was failed to detect the barcode regions properly. So again we trained our model with lots of images. The detector provided with the satisfactory result of locating multiple barcodes from an image than the previously trained model. We went above architecture to train the model for our purpose. The training processes which are needed to implement our method, are described in the following sections.

5.3.1 Collecting Data

The first step is collecting images for the projects. Straight computer generated barcode is not used as it misses out various conditions in real time image. Various conditions were kept in mind while creating the dataset. Wide angle, different angles camera position varies. Lighting conditions- images in proper lighting, images in darker lighting conditions, reflections due to light. Background conditions- white background in darker background, overlapping barcodes, scaling etc.

Since images were taken in IPhone, the original images were pretty big like 1920 × 1090. This would require a lot of memory so images were resized to 784 × 588 to keep an aspect ratio. This was done by using PIL. Sample images are given below.

Figure 5.1: Collection of Images

5.3.2 Annotating the images

After successfully collecting data, it should be converted into a format that our model understands Labelimg was used to annotate the images. This is a very useful tool and annotations are created in the Pascal VOC format which is useful later on. It is written in Python and uses Qt for an interface. Annotation is needed because we identify the *xmin*, *ymin*, *xmax*, and *ymax* for custom object and pass that to the model along with the image for training.

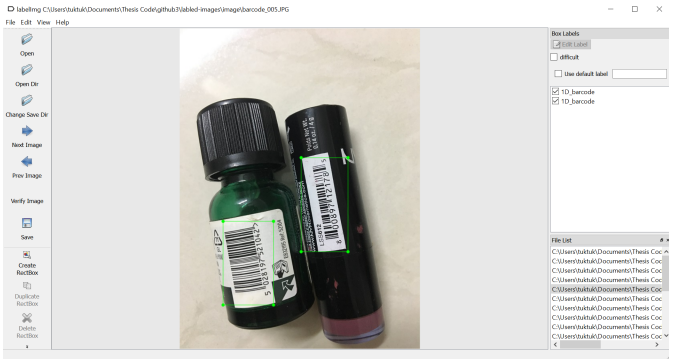

Figure 5.2: Annotation of images

5.3.3 Generate Training Data

The next step is generating TFR records after completing the task of image labeling. We are using Tensorflow API to train. Generating TFR records is necessary because TensorFlow API needs datasets to be in TFR file format. This part is the most sensitive part. TFRecords serve as input data to the TensorFlow training model. Our model uses the *xml_to_csv.py* and *generate_tfrecord.py* scripts.

5.3.4 Creating a model config file

Tensorflow API is equipped with a lot of existing models for object detection purposes. When the TFR datasets are created, then comes a decision-making step. It is time to decide which existing model to use. Using an existing model is beneficial because most of the features that are learned by CNN are often object agnostic and fine-tuning. Enforcing an existing model is usually an easy and accurate process. The API provides 5 different models that provide a tradeoff between speed of execution and the accuracy in placing bounding boxes. The models used in tensorflow object detection API is given in following figure:

Model name	Speed	COCO mAP	Outputs
ssd_mobilenet_v1_coco	fast	21	Boxes
ssd_inception_v2_coco	fast	24	Boxes
rfcn_resnet101_coco	medium	30	Boxes
faster_rcnn_resnet101_coco	medium	32	Boxes
faster_rcnn_inception_resnet_v2_atrous_coco	slow	37	Boxes

Figure 5.3: Model Configuration Details

For executing our idea, *the faster_rcnn_resnet101* which was trained on coco dataset was used. This model was prioritized over *ssd_mobilelenet* as *ssd_mobilelenet* is frequently used to train in low computational devices. Another fact about *ssd_mobilenet* architecture gives faster detection but much less accuracy. The *faster_rcnn* architecture gives slower detection but with more accuracy. As accuracy was preferred over time *faster_rcnn* model was chosen because in reasonable training time it gives satisfactory accuracy.

5.3.5 Training the model

The training was done locally on a GPU(gtx 1050). Training and testing happen at the same time - the scripts in the API run a testing step after every training step. During hours of the training process tensboard was initiated to monitor performance. Initially, each step

of training reports the loss. Preliminary, loss will start high and get lower and lower as training progresses. For our training on the *Faster-RCNN-Inception-V2* model, it started at about 3.0 and quickly dropped below 0.8. Ideally, the model should continue to train until the loss consistently drops below 0.05.

Figure 5.4: Classifier performance in basis of loss in earliest hour of training

Figure 5.5: Classifier performance in basis of loss in latest hour of training

5.3.6 Testing the model

After all these steps the object detection classifier is ready to use. The model's result is analyzed by giving different types of images. Images that classifier has not seen in the training phase, as well as images, are seen in the training phase. While testing the model on

different pictures, we have got good accuracy for the detection. After running the detector it provides the detected barcode region (bounding box). The bounding box (detected region) comes along with the probability of being in a specific class while testing by single run interface. So our classifier located the region and the probability of being in that region was always above 90%.

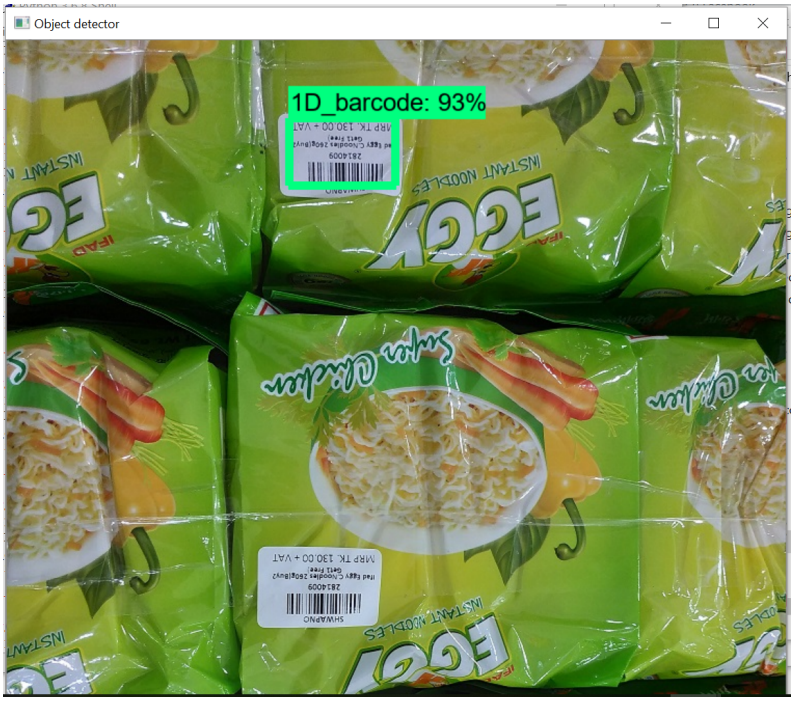

Figure 5.6: Visual representation of model detection performance increased on the same image after being finally trained

In Figure 5.6, the detector performs the detection on a new image. The detector is successful to locate one barcode region it is with the probability of 93%.

In the following figure, the detector performs the detection on an image consisting of multiple barcodes overlapping. The detector is successful to locate all the barcode regions it is with the probability of 99%.

Figure 5.7: Visual representation of model detection performance in the complex background when barcodes are overlapping when the model was finally trained

The detector performs the detection on an image with multiple barcodes of tiny objects which is shown in Figure 5.8. The detector is successful to locate all the barcode regions it is with the probability of 99%.

Figure 5.8: Visual representation of model detection performance on tiny objects after model was finally trained

5.4 Decoding of Barcode

The step comes afterward of detection is decoding. When an image is given to the detector it locates the barcode regions and detected regions are cropped. These cropped images are sent to the Zbar library for decoding. ZBar barcode reader is useful as it can read barcodes from different types of sources. The decoding process is done on single barcodes because Zbar library shows the satisfying result on single barcode decoding where it often fails to decode multiple barcodes. The decoding process is implemented by calling decode function that is a built-in function in python. We have shown "Type" and "Value" of the barcode(s) in the decoding process.

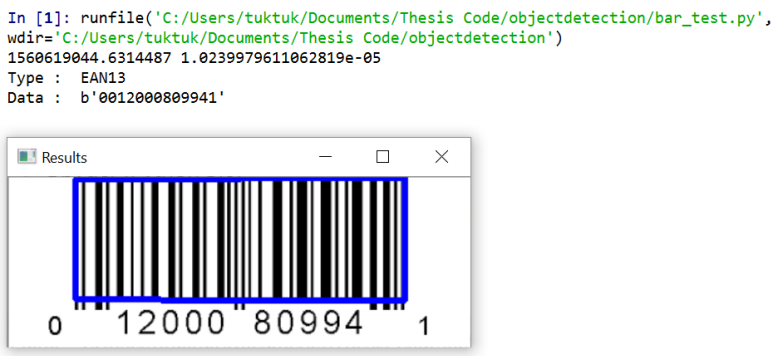

Figure 5.9: Decoding portion by applying pyzbar library

In figure 5.9, the demo of decoding of a single 1D barcode is shown. As we can see in the image decoding shows the type as well value is shown.

To summarize the implementation, a deep learning-based detector is built. The detector detects multiple barcodes at a time. After that, the located regions are decoded separately. In decoding, it shows the value of the barcode which is needed to further proceed things.

Chapter 6

Experimental Results

In this chapter, we will analyze our experiment's result and dataset. Our proposed methodology has two parts: detection and decoding. For detection, we have built a deep learning-based detector. For the detectors' training purpose, we have created a new dataset. An overview of our dataset is given in the following section 6.1. Primarily the detector was trained on fewer images and the detectors' efficiency is shown in images in section 6.2.1. As the result was not up to the mark the detector was trained on more images around 300 images. The models' performance on the training phase is in section 6.2.2. When the detector was finally trained, output efficiency is discussed in length in the section 6.2.3. In section 6.4, decoding accuracy is discussed. Decoding accuracy was checked using the Arte-lab dataset. Analysis of the detectors' performance is showed via graphs and images.

6.1 Dataset

As discussed earlier, to execute the first step of our idea, we need proper data collection. It is a necessary step because until we achieve proper data collection, we will find ourselves constantly coming back to this step. Data can be collected by two sources; one is a primary source the other one is a secondary source. Collecting data from a secondary source means reusing data. Reusing data means that data has already been collected and experimented by other sources, such as data used by certain scientific research. For detecting purpose, we used the primary sources' dataset. And for decoding purpose, we used secondary dataset.

We have obtained our dataset because there was no suitable dataset available to fulfill our training purpose. The only available dataset is the Arte-lab dataset which is used for decoding. This dataset is not suitable for our detector training purpose because there is a lacking of multiple barcodes. Among the 1D single barcodes on the dataset, only the barcode regions were focused. There was a lacking of complex backgrounds too. Complex background

is necessary for training because it will be beneficial for the detector to learn better. It is difficult to find barcode exact regions while detecting multiple barcodes.

Barcode dataset could be created using dummy barcodes or computer generated barcodes. Computer generated barcodes are not suitable to serve our purpose. Firstly, images have many constraints that dummy barcodes do not have. Another fact is that the Faster RCNN models learn by selecting regions within the image. That means it needs to be compared within the region where the barcode is located.

While creating our dataset we wanted the images consisting of both 1D and 2D barcodes. The availability of 2D barcodes is scarce because 2D barcodes are mostly used in digital advertising, manufacturing, and logistics. We were not able to add 2D barcodes to our dataset. Our dataset has three hundred images consisting of 1D barcodes. In table 6.1, the detailed description of object shape is given.

Table 6.1: Dataset for detecting and decoding barcodes

Product	Product type	Product Shape	Barcode Type
Cosmetics	Facewash	Quadruple	EAN-13(1D)
	Facewash	Cylindrical	EAN-13(1D)
	Lipstick	Cylindrical	EAN-13 (1D)
	Lipstick	Rectangular	EAN-13 (1D)
	Lotion	Rectangular	EAN-13 (1D)
	Lotion	Curved rectangular	EAN-8 (1D)
	Cream	Round	EAN-13 (1D)
	Cream	Curved Quadruple	EAN-13 (1D)
	Oil	Cylindrical	UPC-A (1D)
	Oil	Round	EAN-13 (1D)
Food	Spices	Curved Rectangle	EAN-13 (1D)
	Can	Cylindrical	EAN-13 (1D)
	Chips	Square	EAN-8 (1D)
	Chocolate	Rectangle	EAN-13 (1D
Stationary	Book	Square	Code-39 (1D)
		Rectangular	Code-39 (1D)
	Marker	Cylindrical	UPC-A (1D)
	Stapler	Rectangular	EAN-13 (1D)
Beverage	Can	Cylindrical	EAN-13 (1D)
	Bottle	Rectangular curved	EAN-13 (1D)
	Packet	Cubic	EAN-13 (1D)
Electrical Accessories	Mobile packaging	Rectangle	EAN-13 (1D)
	Speaker packaging	Square	EAN-13 (1D)
	Headset packaging	Square	EAN-13 (1D)

Though our dataset does not have 2D barcodes, still we can say we have achieved proper data collection. There are several factors which were kept in mind while creating the datasets. Firstly, the shapes of objects were considered to create the dataset. It is easy to locate barcode from regular shape objects like square or rectangle, but it is challenging to find barcodes in round or cylindrical objects.

After that, the area that we focused, is the size of the objects. It is easy to find barcodes in big objects but it is challenging to locate the barcodes in tiny objects. The images were taken from a different angle of barcodes located. The color of barcodes was an issue in creating the dataset. As barcode is not only black and white stripes anymore, there are blue-white, red-yellow different types of barcode color combinations. There are lighting conditions such as: images in proper lighting, images in darker lighting conditions, reflections due to light. And the background conditions are like: a white background in a darker background, overlapping barcodes, scaling, etc. Different categories of objects were taken because it will be useful for analyzing barcodes to do various classifications.

The device that was used to take the images was the iPhone 6s. The original images were big in resolution which was about 1920×1090. This would require a lot of memory so the images were resized to 784×588 to keep an aspect ratio.

6.2 Detection process

6.2.1 Model's result after training in fewer images

Our main focus was to identify multiple barcodes at a time. Firstly, the model was trained on very fewer images around (50 images) to see if the barcode region is correctly recognized using deep learning. The detector was not able to locate properly. There were false positive regions included in the bounding box.

As we can see in the image, when the model was trained on fewer images the object detector was unsuccessful in providing the desired result. The detector located the regions but there were regions included which do not have barcodes, it is including the background also.

While training the model in fewer images it was showing an error of insufficient images as it could not learn properly to find the barcode regions.

Figure 6.1: Visual representation of the object detector output (1)

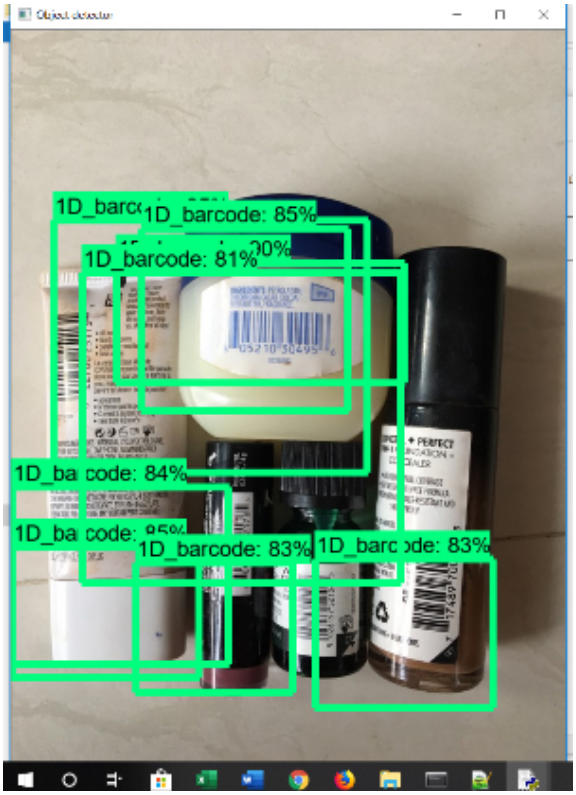

Figure 6.2: Visual representation of the object detector output (2)

Later the model was trained with a lot of data. A large dataset was fed to the model to learn properly so that the detector could find corrected regions. The false detected region was reduced to a lot amount. As mentioned earlier, around 300 images were taken to train and test the purpose.

6.2.2 Model's performance while training on large dataset

Learning rate is a hyper-parameter that controls how fast we are adjusting the weights of our network with respect the loss gradient. Instead of using a fixed value for learning rate and decreasing it over time, if the training does not improve loss anymore, TensorFlow API changes the learning rate every iteration according to some cyclic function. Each cycle has a fixed length in terms of number of iterations. This method lets the learning rate cyclically vary between reasonable boundary values.

Figure 6.3: Learning rate graph while training in TensorFlow

Classification/ localization loss values are the result of loss values are the result of loss functions and represent the price paid for inaccuracy of predictions in the classification/localization problems.

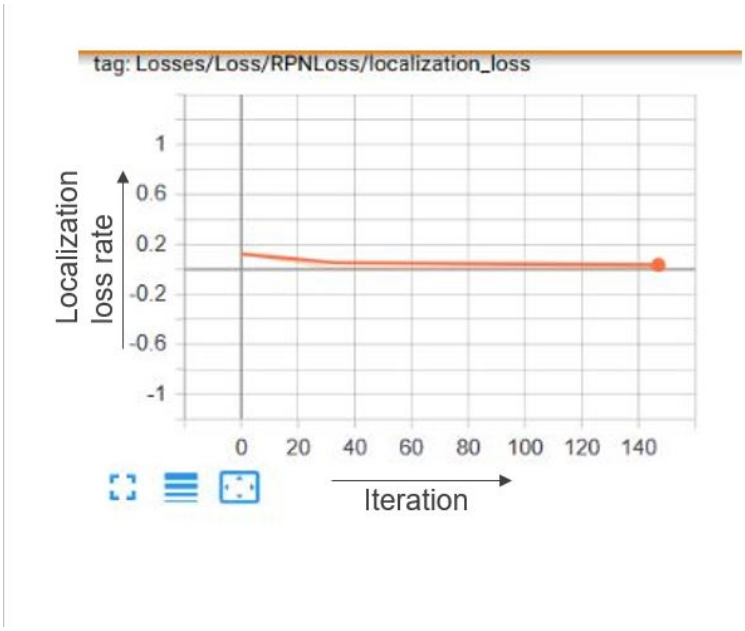

Figure 6.4: Localization loss while training in the earliest hour of training

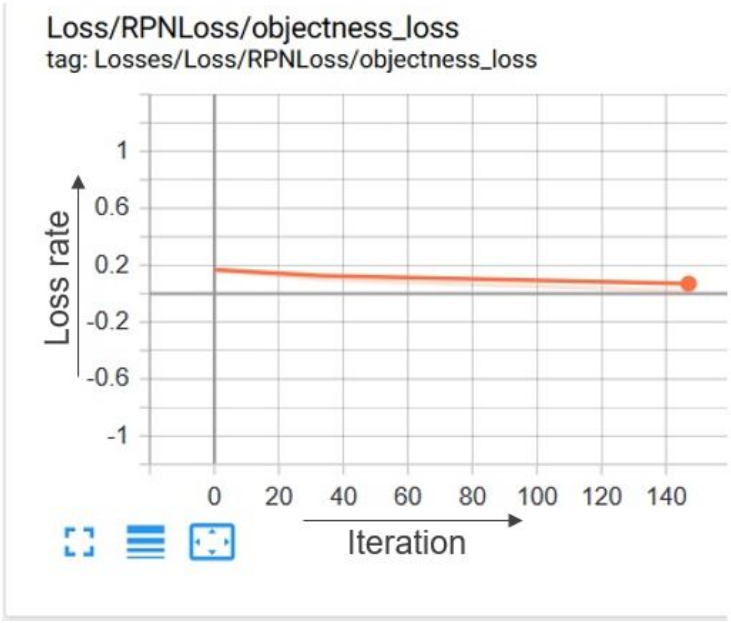

Figure 6.5: Objectness loss while training in the earliest hour of training

Each step of training reports the loss. It will start high and get lower and lower as training progresses.

Allowing the model to train until the loss consistently drops below 0.05. As in the following figure, it is seen after 6000 iteration the loss nearly drops to 0.

Figure 6.6: Classification loss while training in the latest hour of training

Figure 6.7: Total loss after training is completed

One important graph is the Loss graph, which shows the overall loss of the classifier over time. In above figure we can see the total loss being optimized after the model is finally trained that being loss being totally zero.

6.2.3 Model's result after training in large image dataset

All the images that are seen is tested by using single run interface. It means the output is checked by giving one image as input and the output is shown with the detected region along with probability.

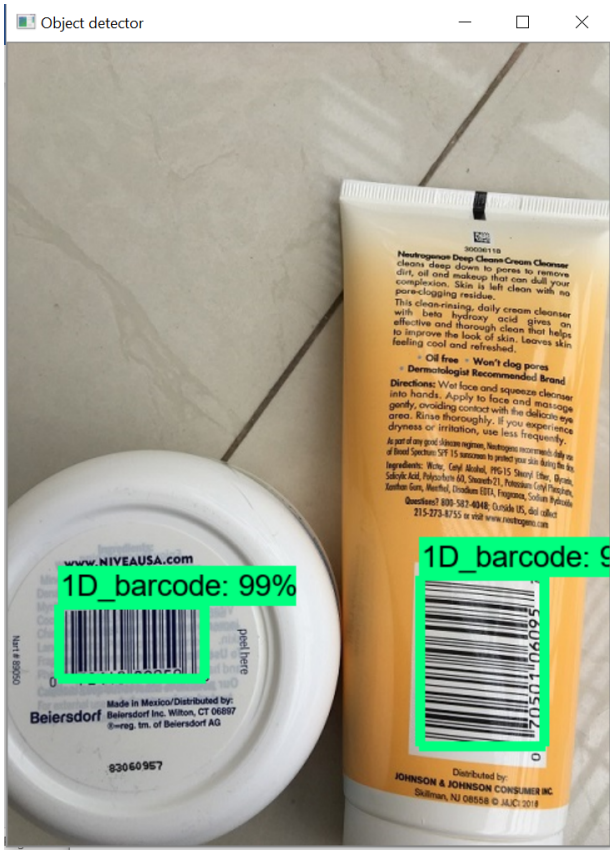

Figure 6.8: Output of the object detector

This image shown in Figure 6.7, was the part of initial training testing dataset. It was taken on white background with proper lighting. Multiple barcodes were detected and the detection region was in the probability of 99%.

And Figure 6.8 shows that the barcode regions was detected for different products. Again there was false positive regions in the background in which barcode was wrapped around was inside the regions. The interesting thing to notice the detector can even find barcode in the reflection even.

Figure 6.9: Resultant detected regions after training on large dataset (1)

In Figure 6.9, the picture was taken in a super shop, to get the exact lighting that is available on super-shops. In this figure there was objects were overlapping. False positive region is there but still all the barcode regions were perfectly recognized by the object detector.

Figure 6.10: Resultant detected regions after training on large dataset (2)

Again this is another image (shown in Figure 6.10) which is taken in real super-shop. In this figure there was objects were in angle with the camera. Camera is not making normal with the objects. The background is more complex than before.

Figure 6.11: Resultant detected regions after training on large dataset (3)

As mentioned in the proposal detecting tiny object's barcode was a concern. In figure 6.11, it was an demonstration of detecting tiny object's barcodes. All the barcodes are perfectly located by the detector.

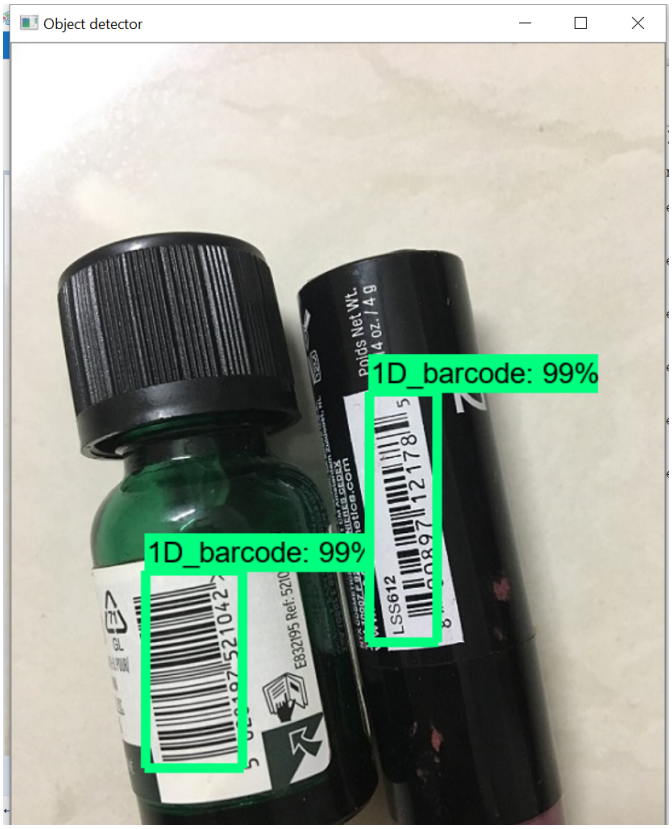

Figure 6.12: Resultant detected regions after training on large dataset (4)

Figure 6.13: Resultant detected regions after training on large dataset (5)

After that, we wanted to see how the detector works in detecting barcode of tiny object in different shape. We have taken different shape and as well as increased the number of

tiny objects which is shown in figure 6.12. All the barcodes are perfectly detected by the detector.

Figure 6.14: Resultant detected regions after training on large dataset (6)

Next we wanted to see how many barcodes can be detected at a time. In figure 6.13, there was a combination of different shape of objects and different color combinations of the barcodes. There was variation of size of the objects. All the barcodes were perfectly detected by the detector.

6.3 Decoding process

In traditional scanning system, Barcode scanners read 1D barcodes horizontally. 1D laser barcode scanners are the most commonly used scanners, and typically come in a "gun" model. These scanners do not need to be in direct contact with the 1D barcode to work properly, but typically need to be within a range of 4 to 24 inches to scan.

In this portion, we will show the detected regions which were cropped and sent to decode using pyzbar library.

Figure 6.15: Cropped region of barcode

Figure 6.16: Results after decoding of barcode

1D barcodes are dependent on database connectivity to be meaningful. If you scan a UPC code, for instance, the characters in the barcode have to relate to an item in a pricing database to be useful. These barcode systems are a necessity for large retailers, and can help increase inventory accuracy and save time.

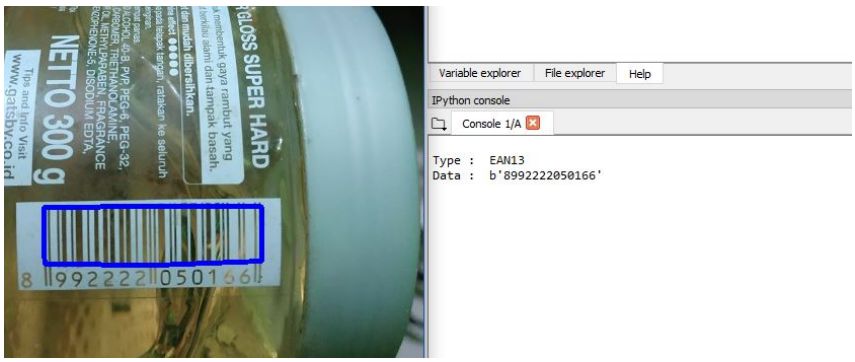

Figure 6.17: Detecting and decoding single 1D barcode from an image (1)

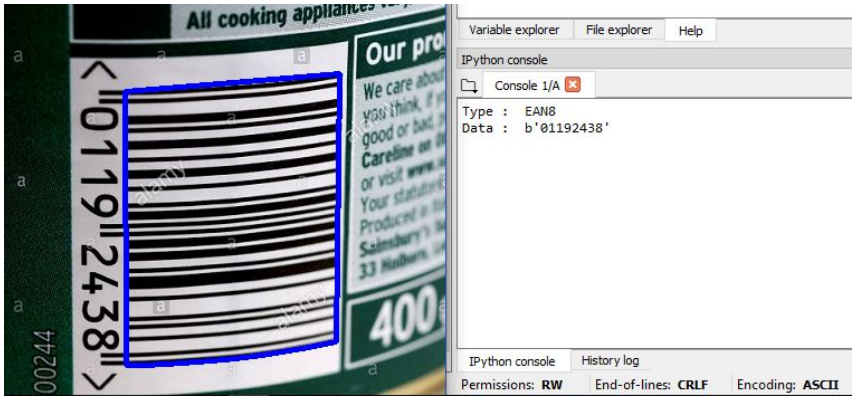

Figure 6.18: Detecting and decoding single 1D barcode from an image (2)

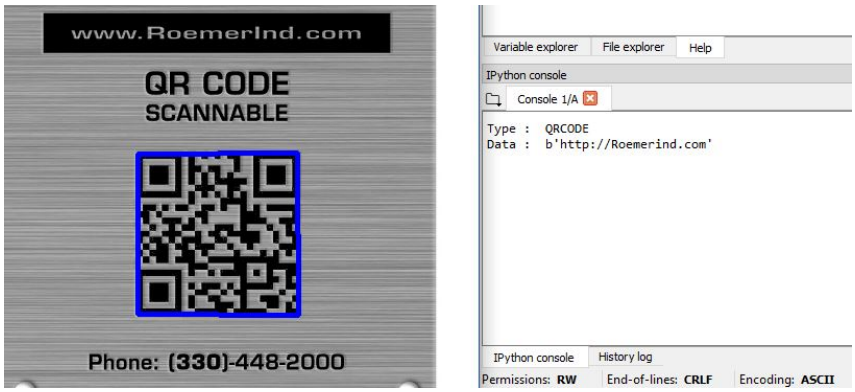

Figure 6.19: Detecting and decoding single 2D barcode from an image

Last of all, we can ensure that our method will work for detecting and decoding single 1D and 2D barcode, multiple 1D barcodes from a snap with proper resolution. The simple and complex background does not matter to detect and decode the barcodes.

6.4 Single 1D Barcode Decoding Accuracy Checking

Accuracy checking:

Table 6.2: Arte database: Resolution vs Accuracy

Resulation	Accuracy
1600 × 1200	89.90 percent
784 × 588	71.60 percent

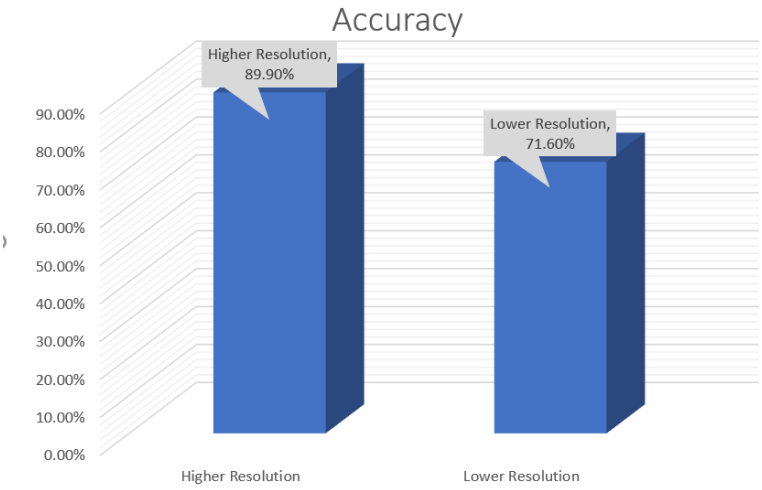

Figure 6.20: Decoding accuracy rate through resolution

Checking time:

Table 6.3: Arte database: Resolution vs Time

Resulation	Time
1600 × 1200	4mins 48 seconds
784 × 588	2 mins 10 seconds

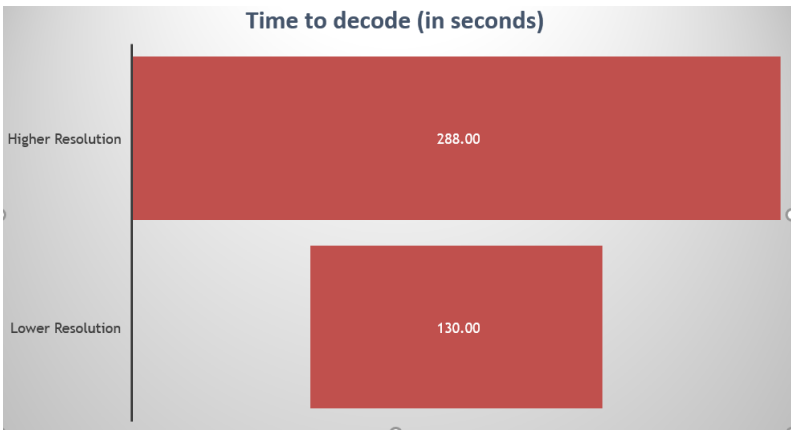

Figure 6.21: Decoding time through resolution

We worked with arte-lab dataset it was here secondary dataset was used. It was utilized for checking the accuracy of decoding 1D barcodes. Accuracy was measured in terms of how many images were decoded by the pyzbar library. It took 4 minutes 48 seconds for the datasets to output the results. Then we changed the resolution in order to check for the accuracy in lower resolution. It took 2 minutes 10 seconds to decode the whole dataset but the accuracy decreased to 71.60%. Detection was possible in the low light condition of the image.

In cases of barcodes located in different angles it failed to decode the barcodes. In some barcodes in complex background ZBar library could not perform the decode operation. So we have thought to use our fabricated dataset to improve the detection and decoding accuracy within less time. The optimal resolution is about 800×600. The graph in Figure 6.18 and 6.19 along with the tables has shown the result.

Arte-lab whole image dataset was taken on normal mobile phone. Now the smart phone's sensors are much superior in the quality so created our own dataset just to test the ZBar librarys' performance. Our dataset was much smaller than the arte-dataset. It showed much better result.

And finally we have compared our system with the detection process of a supershop. We got better result for the detection. Our system is about three times faster than their detection process. Because we can detect many products at a time which is a plus point for us.

Chapter 7

Conclusion and Future Work

7.1 Limitation

We have achieved success in detecting single and multiple barcodes. We also accomplished decoding 1D barcodes as well as 2D barcodes. We obtained 71% accuracy while decoding single 1D barcodes using the Zbar library. This accuracy rate tells us the library's performance was not up to the mark in decoding in some cases. The library could not decode multiple barcodes. Deep learning approach was used in detecting multiple 1D barcodes. Afterward for decoding Zbar library was used so that we can get the accuracy of decoding single barcodes. From discussing what we accomplished, it is noticeable that there are some limitations in our current method like detection and decoding works separately. There is no graphical user interference. We will try to overcome all the limitations in the future.

7.2 Future Work

We have mentioned that detection and decoding works disjointly and lots of manual works needed to be done. Our first target would be linking the two processes. Besides that, there is a scope of a lot of work. We will be focusing on building a better detector than the current one. Building a better detector means reducing the false detected regions as well as detecting all the barcodes. Another thing we need to overcome is to handle detection of the region perfectly in case of angle invariant. We are cropping the region manually, but in the case of automating, we have to set the threshold value so that the detector can crop the detected regions also.

Our target is to build a better classifier than the current one. Our current classifier can detect only one type of barcodes and that is 1D barcodes. Our goal is to build a classifier

that can classify different types of barcodes and can also detect multiple 1D and 2D barcodes at the same time. Further building a classifier that can not only detect 1D and 2D barcode but also classify in their types like in 1D barcodes. We can compare our detectors' timing efficiency with other models. Another goal is to make a real-time application, which will work with videos. The most crucial challenge would be building a detector with the help of deep learning which can decode. In case of 2D barcode decoding it will be challenging to teach the detector which pattern represents what. So the main target is to create a model that can detect and decode as well.

7.3 Conclusion

Barcode is used for all kinds of products and it represents the product specifications. The detection of barcode plays an important role to collaborate the products and match the product id to get trifles about the product. Normal barcode scanners take much time to detect the barcodes one by one. Also, these scanners are very costly and need user guidance. If we could make a user-friendly system that can detect and decode multiple barcodes simultaneously, it will be beneficial for every sector.

In the shopping system as well as the business system, barcoding technology is a must for storing the product information. Hopefully, our idea will help especially those sectors where multiple barcode detection and decoding process would make a system faster than before ever. The optimal goal is to help to increase productivity, lessen user guidance, reduce time, open a new window of comparison to reduce the confusion of consumers and lead a better life.

References

[1] X. Li, Z. Shi, D. Guo, and S. He, "Reconstruct argorithm of 2d barcode for reading the qr code on cylindrical surface," in *Anti-Counterfeiting, Security and Identification (ASID), 2013 IEEE International Conference on*, pp. 1–5, IEEE, 2013.

[2] "1d barcode structure." http://www.dataid.com/whatsabcdetail.htm. Accessed: 2019-05-27.

[3] "Faster rcnn model." https://zhuanlan.zhihu.com/p/31426458. Accessed: 2019-06-17.

[4] "Feature map model." https://bit.ly/2RtfDjT. Accessed: 2019-06-17.

[5] "Region proposal network (rpn)." https://bit.ly/2Lf7u1z. Accessed: 2019-06-17.

[6] "Vgg 16 network." https://zhuanlan.zhihu.com/p/41423739. Accessed: 2019-06-17.

[7] "R-cnn, fast r-cnn, faster r-cnn, yolo - object detection algorithms." https://bit.ly/2zznZRA. Accessed: 2019-06-16.

[8] "What is barcode." http://www.dataid.com/whatisbarcode.htm. Accessed: 2019-06-02.

[9] "Barcode: The ultimate guide to barcodes." http://www.waspbarcode.com/buzz/barcode. Accessed: 2019-05-30.

[10] "Definition of barcode." https://quizlet.com/111262719/bus-tech-or-codes-and-apps-flash-cards. Accessed: 2019-06-19.

[11] "Difference between 1d and 2d barcode scanning." https://bit.ly/2x7JKnx. Accessed: 2019-06-02.

[12] R. Adelmann, M. Langheinrich, and C. Floerkemeier, "Toolkit for bar code recognition and resolving on camera phones - jump starting the internet of things.," pp. 366–373, 01 2006.

[13] T. R. Tuinstra, "Reading barcodes from digital imagery," 01 2006.

[14] S. Wachenfeld, S. Terlunen, and X. Jiang, "Robust recognition of 1-d barcodes using camera phones," in *2008 19th International Conference on Pattern Recognition*, pp. 1–4, Dec 2008.

[15] S. S. Upasani, A. N. Khandate, A. U. Nikhare, R. A. Mange, and R. Tornekar, "Robust algorithm for developing barcode recognition system using web-cam,"

[16] C.-S. Tseng, K.-T. Wang, M.-C. Wu, N.-Y. Cheng, and J.-H. Wang, "Retrospective tracking for barcode reading," in *Industrial Informatics (INDIN), 2010 8th IEEE International Conference on*, pp. 114–119, IEEE, 2010.

[17] D. Bradley and G. Roth, "Adaptive thresholding using the integral image," *Journal of graphics tools*, vol. 12, no. 2, pp. 13–21, 2007.

[18] F. Chang and C.-J. Chen, "A component-labeling algorithm using contour tracing technique," in *null*, p. 741, IEEE, 2003.

[19] Y. Zhang and T. Lu, "A fast color barcode detection method through cross identification on mobile platforms," in *Document Analysis and Recognition (ICDAR), 2015 13th International Conference on*, pp. 416–420, IEEE, 2015.

[20] S.-C. Lin and P.-H. Wang, "Design of a barcode identification system," in *Consumer Electronics-Taiwan (ICCE-TW), 2014 IEEE International Conference on*, pp. 237–238, IEEE, 2014.

[21] R. C. Gonzalez, R. E. Woods, *et al.*, "Digital image processing," 2002.

[22] A. Zamberletti, I. Gallo, and S. Albertini, "Robust angle invariant 1d barcode detection," in *Pattern Recognition (ACPR), 2013 2nd IAPR Asian Conference on*, pp. 160–164, IEEE, 2013.

[23] C. Galamhos, J. Matas, and J. Kittler, "Progressive probabilistic hough transform for line detection," in *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, vol. 1, pp. 554–560, IEEE, 1999.

[24] M. Katona and L. G. Nyúl, "A novel method for accurate and efficient barcode detection with morphological operations," in *Signal Image Technology and Internet Based Systems (SITIS), 2012 Eighth International Conference on*, pp. 307–314, IEEE, 2012.

[25] I. Zafar, U. Zakir, and E. Edirisinghe, "Real time multiple two dimensional barcode reader," in *Industrial Electronics and Applications (ICIEA), 2010 the 5th IEEE Conference on*, pp. 427–432, IEEE, 2010.

[26] "Faster rcnn object detection." https://towardsdatascience.com/faster-rcnn-object-detection-f865e5ed7fc4. Accessed: 2019-06-12.

[27] "A deeper look at how faster-rcnn works." https://bit.ly/2LaYGtk. Accessed: 2019-06-17.

[28] "Image classifier - cats vs dogs." https://bit.ly/2BhsoaF. Accessed: 2019-06-13.

[29] "Real-time barcode detection in video with python and opencv." https://www.pyimagesearch.com/2014/12/15/real-time-barcode-detection-video-python-opencv. Accessed: 2019-06-12.

[30] D. Kold Hansen, K. Nasrollahi, C. B. Rasmusen, and T. Moeslund, "Real-time barcode detection and classification using deep learning," pp. 321–327, 01 2017.

[31] "An introduction to neural networks for beginners." https://www.coursehero.com/file/40556871/An-introduction-to-neural-networks-for-beginnerspdf. Accessed: 2019-06-14.

Generated using Undegraduate Thesis LaTeX Template, Version 1.4. Department of Computer Science and Engineering, Ahsanullah University of Science and Technology, Dhaka, Bangladesh.

This thesis was generated on Monday 1st July, 2019 at 6:51pm.