Multi-lingual Text Identification and Recognition: A Deep Learning Approach

Thesis submitted in partial fulfillment of the requirement for the award of the degree of

Master of Engineering in Computer Science and Engineering

In the Faculty of Engineering and Technology

Jadavpur University

By

Biswajit Das

Exam Roll No **:** **M4CSE22012**

Registration No**:** **154136 of 2020 - 21**

Under the Guidance of

Prof. (Dr.) Subhadip Basu

Department of Computer Science and Engineering

Jadavpur University

Kolkata - 700 032

2022

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
FACULTY OF ENGINEERING AND TECHNOLOGY
JADAVPUR UNIVERSITY

CERTIFICATE

I hereby recommend that the thesis entitled "**Multi-lingual Text Identification and Recognition: A Deep Learning Approach**" prepared under my supervision by **Biswajit Das,** Exam- Roll No**: M4CSE22012**, be accepted in partial fulfillment of the requirements for the degree of **Master of Engineering in Computer Science and Engineering** of **Jadavpur University, Kolkata**.

Supervisor

Prof. Subhadip Basu
Department of Computer Science & Engineering,
Jadavpur University, Kolkata - 32

Prof. Anupam Sinha
Head of the Department,
Computer Science & Engineering,
Jadavpur University, Kolkata- 32

Prof. Chandan Mazumdar
Dean,
Faculty of Engineering & Technology,
Jadavpur University, Kolkata- 32

JADAVPUR UNIVERSITY
FACULTY OF ENGINEERING AND TECHNOLOGY

CERTIFICATE OF APPROVAL*

The foregoing thesis "**Multi-lingual Text Identification and Recognition: A Deep Learning Approach**" at instance is hereby approved as a creditable study of an engineering subject carried out and presented in a manner of satisfactory to warrant its acceptance as pre-requisite to the degree for which it has been submitted. It is notified to be understood that by this approval, the undersigned do not necessarily endorse or approve any statement made, opinion expressed and conclusion drawn there in but approve the thesis only for the purpose for which it has been submitted.

Final Examination for the Evaluation of Thesis

Board of Examiners

_____

_____

(Signature of Examiners)

* Only in case thesis is approved

ACKNOWLEDGEMENTS

I express my profound gratitude and sincere thanks to **Prof. Subhadip Basu** for his valuable suggestions, guidance, constant encouragement and intent supervision at every stage of thesis work. It has been a great learning process for me. It is in his association that gave me many opportunities to enhance my skills and knowledge.

I am grateful to **Prof. Mita Nasipuri**, formar Coordinator of CMATER Laboratory, for allowing me to use the computing facilities of "Center for Microprocessor Application for Training Education and Research", "Project on Storage and Understanding of Video for Multimedia" laboratory, in the Department of Computer Science and Engineering, Jadavpur University.

I am also thankful to my senior **Mr. Neelotpal Chakraborty**, Phd. Scholar, JU, Dept. of CSE, JU for providing his immense support from scratch and encouragement that I received to complete my thesis.

Last but not the least, I express my gratitude to my friends and family for their constant support and unfailing guidance in whatever I did.

Date :

Place : Kolkata

BISWAJIT DAS

Signature:

M.E. (C.S.E)

Roll No: 002010502012

Abstract

In this thesis, an attempt has been made to solve the problem of text detection and identification for multi-lingual text on natural images and applied a pre-trained model, Tesseract OCR to recognize these texts. A method for detecting and classification of the multilingual text using You Look Only Once (YOLO), a Deep Learning approach and recognition of the text using Tesseract OCR from natural scene images is proposed. To accomplish this task, YOLO version-3 has been used which is immensely fast and give accurate result. The YOLOV3 uses the Darknet-53 and has an overall 53 conventional layers. This model not only predict the location and make bounding boxes over the texts but also predict the class level of the bounding box. The convolutional networks are first trained on Custom dataset with smaller number of images. Then a standard dataset, MITDI, containing 400 train natural scene images and 400 train born digital images, is used to effectively train the model. Using the trained model, the text regions are successfully extracted from the input image and the proposals obtained are labelled with the language class of the text within it. Then, Optical Character Recognition (OCR) engines are used to recognize the text in a specific language. Tesseract which has been implemented using a Long Short-term Memory (LSTM) based recognition engine, performed well on the natural scene images. Multiple applications can be realized, ranging from human computer interaction for visually impaired person, context extraction from image, autonomous machines etc.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

As we know, natural scene text images are surrounded around us and detecting text over such images is really a challenging research problem. Many real time applications can be made using scene text recognitions, such as: text translation, navigation system for visually-impaired, mobile applications etc. For these reasons, recognizing textual content from scene images are getting immense interests from the research community [1]. A traditional optical character recognition (OCR) problem, where the texts are extracted against simple backgrounds, is easy to recognize the characters using existing/pre-trained printed character recognition models, like Tesseract. But in reality, an outdoor scene image may consist of several complex background such as car, house, pedestrian etc. and localizing and/or segmenting only text regions take large amount of attention. Therefore, the general strategy is to localize the correct locations of texts in images, and subsequently apply OCR for text recognition and eventually develop practical applications [2].

1.1 Overview

The process of Scene Text Localization and Recognition search all areas in an image or a video that would contain text, make boundaries of the text areas and output a sequence of characters associated with its content. They are used to process images and videos taken by a digital camera or mobile phone and to read the content of each text area into a digital format. After that, the character sequences are processed to make an Index based image retrieval application[3]. Natural Scene Text Localization and Recognition is an open and complex

problem of computer vision, because text typically takes only a small fraction place of the image[4], it has non-uniform background, it suffers from noise, blur and perspective effects need to be taken for account. Moreover, real-world texts are often written in different fonts and languages. All the factors make the Scene Text Localization and Recognition problem significantly harder and standard printed document recognition (OCR) methods and applications cannot be used[5]. The goal of the thesis is to advance the state of the art by introducing methods specialized in text localization and recognition, without constraining the methods to a particular domain of text, a font, a script, or a type of scene the text is captured[6].

Methods for scene text localization and recognition are very useful, because text captures important information about the scene and they are also an important component of general image understanding. For example, in the ICDAR dataset for text detection, there are over 3000 multilingual text instances which makes it the most frequent class. There are also many possible applications of scene text localization and recognition with some particular examples listed below[7].

Figure 1.1: Multilingual Texts Consist of Complex Background

1.2 Motivation

Natural scene texts recognition is in the forefront of Artificial Intelligence today. It is however far from perfection. Seemingly simple scenarios, such as text detection, text recognition, removing motion blur, etc. and more complex scenarios such as automatic translation, translation system for visually impaired person, and spelling cheaker are applications of scene text recognition/text detection. Digitized images are often represented as a two-dimensional (2D) array of pixels values. Each pixel value which makes up the color scheme of the image is often influenced by an array of factors such as light intensity. Visual scene is projected unto a surface, where receptors (natural or artificial) produce values that depend on the intensity of incident light. These exciting concepts are however hard to implement. Forming an image leads to loss of details of information while collapsing a three-dimensional (3D) image into a two-dimensional image. So many factors are responsible for why text detection is hard. Some of such factors are noise in the image (pixels values that are off from its surrounding pixels), mapping from scene to image etc. A faster object detector, YOLO, was proposed to implement object detection in real-time situation. Our motivation is to apply YOLO to text detection task of natural scene images. We will also be comparing the speed and accuracy of this with an OCR software

1.3 Scope of Present Work

As mentioned earlier, text detection in natural scenes is a difficult task and more complicated than text extraction in document text images, where there is a clear distinction between background and foreground and each character is separated from the context. In natural scenes, text can be appeared in numerous states, dark text in light background and vice versa, with wide variety of fonts, even for characters of the same word, part of words can be overlapped by object of the environment and as a result the detection of these parts can be impossible. Other factors, like camera settings, may cause blurry images or perspective distortions[10]. A major factor that makes the text detection and recognition in natural scenes difficult, are the illumination conditions. The light of the environment may create reflections on the text surfaces, object of the environment may cast shadows on the text surface, and also

the intensity of the objects depends on the light source [11]. So during our present work, we are able to localize more than one language and classify each language.

The most common approach for text detection in natural scenes is the scanning of the input image with a single shot detection of different scales and the evaluation of these patches as text or no-text. This approach requires on average approx. 57 convolution network classifier evaluations per image [3]. Then Dense layer are required at the end of the classifier to detect bounding box and using non-max suppression we get exact same box which is in ground truth value We achieve recognition part using Tesseract tool with the extraction of different color channels followed by two different text detection algorithms, in order to detect the character areas and then the connection of these characters into words. By doing this, we achieve not only to reduce the search space substantially, but also to take advantage of the properties of each color channel, since each one is invariant to different illumination conditions[13]. The main scope of our present work is to make several applications to enhance human-computer interactions.

1.4 Applications

An efficient text detection and text recognition approach has many applications even in our daily life. Can be a useful component of navigation devices when it successfully recognizes the text on the street signs, can be a necessary tool for the blind or the visually impaired people [6], when a URL is detected can be combined with a browser to navigate to this website. A more general use can be achieved with combination with information retrieval systems; the user can retrieve additional information about an entity like a physical person, a city, an event etc.

1.4.1 Helping visually impaired

It is very complex or even impossible for visually impaired or elderly people to read texts, but text often provides crucial information which sometimes cannot even be possibly obtained in any other way, for example – the only way to differentiate between two different types of medicines can be just by reading the label on its packaging. Linking the scene text recognition algorithm with a speech synthesizer and deploying it on a mobile phone gives an

easy-to-use and affordable assistive tool, which is able to help by automatically detecting any text in the video stream and reading it out loud. The method must not only be accurate but also efficient, because text has to be automatically detected directly from the live camera stream[8], as it is not reasonable to expect that visually impaired people would take pictures of text.

1.4.2 Navigation System

Navigation in outside or inside buildings cannot be relied on position information from GPS, so other methods have to be evaluated. One of the options is using business labels and other signs as one of the cues to determine the position, and to transform same applies for self-driving cars a scene text recognition algorithm is therefore required, where GPS and maps are available, but they may not include temporary signs, which are often text -based.

1.4.3 Automated translation

One of the Problem in machine translation applications is the user input. Traditionally, a user has to type in the word/phrase that is translated, however this can be very slow, error-prone and sometimes feels impossible, if the user is not familiar with the alphabet of the text which tried by him to translate. Automated natural scene text recognition overcome the issue by automatically recognizing text in an image taken by the user, possibly with a guidance from the user to select which areas of the image to translate, effectively avoid the need for any manual input.

1.4.4 Indexing and searching image databases by textual content

An arbitrary image or video datasets can be automatically indexed by its textual content so that user can search by text queries, which is the most common input to search engines [9]. An user could find his favorite restaurant in a database such as Google Street View using just a restaurant’s name etc.

1.5 Organization of Thesis Work

Organization of thesis is described below:

Chapter 1: Introduction

Here, a brief explanation about text detection and its background, motivation behind the current work, some basic characteristics of text recognition and finally organization of the thesis is given.

1.1 Overview

1.2 Motivation

1.3 Scope of Present Work

1.4 Organization of Thesis Work

Chapter 2: Literature Survey

This chapter describes some previous works regarding Text detection and recognition using different deep learning approaches.

Chapter 3: Methodology

3.1 Overview

In this chapter, theoretical knowledge required for understanding the methods discussed in the chapter 3 has been provided. Details about machine learning, neural networks, and computer vision have been discussed, followed by the explanation of the Efficient

3.2 Machine Learning

3.3 Computer Vision

3.4 Convolutional Neural Network

3.5 EAST: An efficient and accurate scene text detector

3.6 YOLO Version 3

Chapter 4: Experiment and Results

The overall process of results extraction is discussed module-wise in this chapter.

4.1 Datasets Preparation

4.2 Training Process

4.3 Experiment Results

4.4 Tesseract OCR

4.5 Pipeline

Chapter 5: Conclusion

This thesis presents a YOLO v3 based end-to-end framework for real-time and static natural scene text detection and recognition and evaluates the performance of these algorithms on the detection and recognition using Tesseract-OCR tool.

5.1 Future Work

English, Bengali, Hindi. But multilingual with more numbers of languages for detection as well as language translation with different RNN like LSTM could be great work in the near future

.

Chapter 2

Literature Survey

Natural scene text detection and recognition have been always active research area in computer vision for a very long time. So many inspiring ideas and effective approaches have been investigated[14][3][15][5].

Most of traditional methods for scene text recognition take the bottom-up approach in which individual character is first detected by sliding window, and then combine them for taking the dependence with its neighbors into consideration[6][16]. These methods may fail to detect small characters, and are easily disturbed by background noise, illumination, and low image quality, blurring, noise, etc. Later, top-down methods were proposed, in which text sequences are end-to-end predicted without the single character detection[17]. Recently, there are approaches which target at complex arbitrary-shaped text recognition become dominant. These approaches can be roughly categorized into rectification-based, segmentation based and encoder-decoder with attention-based[8].

Yao C[10] proposed a system by which texts of arbitrary orientations in natural scene images are detected. Their algorithms contain a two-level classification scheme and two sets of features specially created for capturing both the intrinsic characteristics of texts. They made a new dataset, which includes different types of texts in diverse real-world scenarios. They also proposed an algorithm(protocol) for performance evaluation. Experiments on few datasets and their own dataset demonstrate that their algorithm compares

desirably with the state-of-the-art algorithms when handling horizontal texts and they achieves significantly increased performance on texts of arbitrary orientations in complex natural scenes but was not working properly for multilingual text.

Wang T et al. [15] proposed a decoupled attention network (DAN), which decouples the different alignment operation from using old decoding results. DAN is an effective, flexible and robust end-to-end text recognizer, which consist of three components: 1) a feature encoder that features are extracted from the input image; 2) a module of convolutional alignment that performs the alignment operation based on visual features from the encoder; and 3) a decoupled text de-coder that made final prediction by jointly utilizing the feature map and attention maps. Experimental results showed that DAN achieved state-of-the-art performance on multiple text recognition tasks, including online and offline hand-written texts recognition and regular/irregular natural scene text recognition.

Huang Y [4] proposed the effective parts attention network (EPAN) which could attentively highlight the characters region for more precise recognition. EPAN consists of at image encoder and character effective parts decoder (CEPD), and it send-to-end trainable. The former divides the high-dimensional feature map into one-dimensional vectors row-by-row, which are connected to a bidirectional long short-term memory unit to encode contextual information. Subsequently, the CEPD transforms the vectors using a network at each time step to roughly evaluate the position of the character. Then the CEPD used were fine network to generate and asked to gradually localize the precise position of important parts of the current character. Experiments were conducted on various datasets, includingIIIT5K-Words, Street View Text, ICDAR2003, ICDAR2013, CUTE80, Street View Text Perspective, and ICDAR2015, which demonstrated that their proposed EPAN method significantly outperformed or was comparable to existing methods in-terms of lexicon-free word accuracy.

Liu C [5] proposed, a Character-Context Decoupling framework is proposed to solve this problem by separating contextual information and character-visual information. Contextual information can be decomposed into temporal information and linguistic information. Here, temporal information that models character order and word length is

isolated with a detached temporal attention module. Linguistic information that models n-gram and other linguistic statistics is separated with a decoupled context anchor mechanism. Different types of quantitative and qualitative experiments showed that their method achieved slightly improved performance on open-set, zero-shot, and close-set text recognition datasets.

Scene Text Recognition for Indian Language with Transfer Learning and Font Diversity [8], [18], [19]present using additional non-Unicode fonts with generally utilized Unicode fonts to cover font diversity in such synthesizers for Indian languages. The author, also performed experiments on transfer learning among six different Indian languages. Their transfer learning experiments on synthetic images with common backgrounds provide an exciting insight that Indian scripts can benefit from each other than from the extensive English datasets. Their evaluations for the real settings helped us to achieve significant improvements over previous methods on three Indian languages from standard datasets like MITDI, ICDAR MLT-19, and the new dataset containing 400 scene images with 2200 English and 1100 Bengali words. We also used transfer learning in our YOLOv3 model. Further enriching the synthetic dataset with non-Unicode fonts and multiple augmentations helped them to achieve a Word Recognition Rate gain of over33%on the IIIT-ILST Hindi dataset. They also presented the results of lexicon-based transcription approaches for all six languages[20].

Shi B [21] proposed a neural network architecture, which integrates feature extraction, sequence modeling and transcription into a unified frame-work, was proposed. Compared with previous systems for scene text recognition, the proposed architecture possesses four distinctive properties: (1) It is end-to-end trainable, in contrast to most of the existing algorithms whose components were separately trained and tuned. (2) It naturally handles sequences in arbitrary lengths, involving no character segmentation or horizontal scale normalization. (3) It was not confined to any predefined lexicon and achieved good performances in both lexicon-free and lexicon-based scene text recognition tasks. (4) It generated an effective and much smaller model, which was more practical for real-world application scenarios. The experiment was done on standard dataset, including the IIIT-5K, Street View Text and ICDAR datasets, demonstrated the well of the proposed algorithm over

the prior arts. Moreover, their proposed algorithm performed well in the task of image-based music score recognition, which evidently verified the generality of it.

The author [22] showed that a class of non-linear kernel SVMs admit approximate classifiers with run-time and memory complexity that is independent of the number of support vectors. This class of kernels which they refer to as additive kernels, include the widely used kernels for histogram-based image comparison like intersection and chi-squared kernels. Additive kernel SVMs can offer significant improvements in accuracy over linear SVMs on a wide variety of tasks while having the same run-time, making them practical for large scale recognition or real-time detection tasks. They also present experiments on a variety of datasets including the INRIA person, Daimler-Chrysler pedestrians, UIUC Cars, Caltech-101, MNIST and USPS digits, to demonstrate the effectiveness of our method for efficient evaluation of SVMs with additive kernels. Since its introduction, their method had become integral to various state of the art systems for PASCAL VOC object detection/image classification, ImageNet Challenge, TRECVID, etc.

Luo C [9] , proposed a multi-object rectified attention network (MORAN) for general scene text recognition. The MORAN consists of a multi-object rectification network and an attention-based sequence recognition network. The multi-object rectification network was designed for rectifying images that contain irregular text. It decreased the difficulty of recognition and enabled the attention-based sequence recognition network to more easily read irregular text. It was trained in a weak supervision way, thus requiring only images and corresponding text labels. The attention-based sequence recognition network focused on target characters and sequentially outputs the predictions. Moreover, to improve the sensitivity of the attention-based sequence recognition network, they have used a fractional pickup method was proposed for an attention-based decoder in the training phase. With the rectification mechanism, the MORAN could read both regular and irregular scene text. Extensive experiments on various datasets are conducted, which showed that the MORAN achieved state-of-the-art performance.

Ballan L [23] surveyed many methods that have been applied to different video domains of texts, considering edited text videos (i.e. videos that have been created from a

collection of video material, selecting what elements to retain, delete, or combine, like movies) and unedited text videos (i.e. videos that have not been processed and are simply the result of video recording, like surveillance videos). A categorization of events and actions related to different video domains and production methods is provided.

In this [24] paper, authors presented a framework for detection and recognition of textual content in video frames. More specifically, they target cursive scripts taking Urdu text as a case study. Detection of textual regions in video frames is carried out by fine-tuning deep neural networks base object detectors for the specific case of text detection. Script of the detected textual content is identified using convolutional neural networks (CNNs), while for recognition, they proposed an Urdu-Net, a combination of CNNs and long short-term memory (LSTM) networks. Dataset containing cursive text with more than 13,000 video frames was also developed.

The author [25] have trained a large, deep convolutional neural network to classify the 1.2 million high-resolution images in the ImageNet LSVRC-2010 contest into the 1000 different classes. The neural network, which has 60 million parameters and 650,000 neurons, consists of five convolutional layers, some of which are followed by max-pooling layers, and three fully-connected layers with a final 1000-way soft-max. To make training faster, they have used non-saturating neurons and a very efficient GPU implementation of the convolution operation. To reduce overfitting in the fully-connected layers we employed a recently-developed regularization method called “dropout”.

Cheng Z [26] developed the arbitrary orientation network (AON) to directly capture the deep features of irregular texts, which are combined into an attention-based decoder to generate character sequence. The whole network can be trained end-to-end by using only images and word-level annotations. Experiments on various datasets, including the CUTE80, SVT-Perspective, IIIT5k, SVT and ICDAR datasets, showed that their proposed AON-based method achieved the-state-of-the-art performance in irregular datasets, and is comparable to major existing methods in regular datasets.

The following conclusions have been drawn from the results obtained through the literature review:

• From the results of performance of various object detection models on the different dataset, it can be concluded that YOLOv3 models are faster when compared to the Faster R-CNN or SSD or any other mentioned method.

• But if accuracy is given preference over speed, then Faster R-CNN performs better than YOLO, SSD and R-FCN models.

• Faster R-CNN is the most accurate model while using Inception Res-Net, running on static image which satisfies the minimum requirement to perform text detection and recognition but for video image Single Shot Detection models like YOLO performs much better than any other.

• SSD is faster compared to other text detection models but has difficulty in detecting small objects.

• Speed of the Faster R-CNN increases as the number of proposals decrease, also decreasing the accuracy of the model.

• According to Redmond et al. [27], YOLOv3 is able to detect 10 times faster than the state-of-the-art methods. Hence YOLOv3 and its variant have been selected for the experimentation.

Chapter 3

Methodology

3.1 Overview

In this chapter, theoretical knowledge required for understanding the methods discussed in the chapter 3 has been provided. Details about machine learning, neural networks, and computer vision have been discussed, followed by the explanation of the Efficient and Accurate Scene Text detection approach (EAST), Faster R-CNN, YOLOv3 and Tiny-YOLOv3. Related work performed by other researchers related to this area of study has also been presented towards the end of this chapter.

3.2 Machine Learning

Machine learning is one of the applications of Artificial Intelligence (AI) which makes the computers to learn on their own and perform tasks without human intervention [28]. There are numerous applications available of machine learning algorithms in the field of computer vision. With the help of machine learning, formulation of some of the most complex problems have been performed easily. Various computer programs which were previously programmed by humans, sometimes by-hand, are now being programmed without any

human contribution with the help of machine learning[29]. In the recent years, due to remarkable increase in the availability of numerous sources of data and feasibility of computational resources, machine learning has become pre-dominant with many ranges of applications in our daily lives[30].

These ML algorithms help different business problem like classification, Forecasting, Clustering and Associations etc. Based on the methods and way of learning is divided into four types, which are:

- Supervised Learning
- Unsupervised Learning
- Reinforcement Learning

3.2.1 Supervised Learning

Supervised Machine Learning is based on supervision. It means in the supervised learning technique, we train the machines using the "labelled" dataset[31], and based on the training, the machine predicts the output. Here, the labelled data specifies that some of the inputs are already mapped to the output. Moreover, we first train the labelled dataset with given ground truth and then test the model with test datasets.

- ***Categories of Supervised Machine Learning***
  - Classification
  - Regression
- **Some of the popularly utilized supervised learning algorithms are:**
  - Neural Networks
  - Decision Trees
  - Random Forest
  - K-Nearest Neighbors
  - Linear Regression
  - Logistic Regression
  - Support Vector Machines

3.2.2 Unsupervised Learning

Unsupervised leaning is different from the supervised learning technique. The machine is trained using the unlabeled dataset and the predicts the output without any supervision. Algorithms like K-means clustering[32], etc. are some of the common unsupervised learning algorithms.

3.2.1.1 Neural Networks

Artificial neural networks are a popular type of supervised learning Model. A special case of a neural network called the convolutional Neural network (CNN) is the **primary focus of this thesis**. The name 'Artificial Neural Networks' was given to this model because they were developed to imitate the neural function of the human brain. An artificial neural network consists of a set of neurons connected to each other and are grouped into layers to replicate the neural function of out brain. Similar to the neurons in a human brain, the neurons in an artificial neural network function as units of calculation. The connections between neurons are known as 'synapses' which are nothing but weighted values. Therefore, in a simple sense, when an input value is provided at a neuron (x1, x2,), it traverses the synapse, multiplying its value with the weighted value of the synapse (w1, w2,) as shown in the Figure below. Bias 'b' is then added to the summation of these values. This will be the output of the neuron. Since a neuron does not know its boundary, a mapping mechanism is required to map the inputs to the output, known as the 'Activation function' [21]. In a fully connected feed- forward multi-layer network, all the outputs of a layer of a layer of neurons is fed as input to every neuron of the next layer. As a result, some of layers get to process the original input data, while some layers get to process the data that has been obtained from neurons from the previous layer. Therefore, the number of weights of any neuron in the network is equal to the number of neurons in the layer previous to the layer of the neuron [19].

$$Y = \sum_{k=1}^{n} (w_n \times x_n) + \mathrm{b} \tag{1.1}$$

In the above equation, ‘x’ is the input value given at the neuron, ‘w’ is the weighted value of the synapse, ‘n’ is the number of neurons, ‘b’ is the bias and ‘y’ is the output of the network. Therefore, according to the equation (1.1), the value of output ‘y’ is equal to the summation of the product corresponding weights and bias ‘b’.

A multi-layered artificial neural network, as shown above includes three types of layers: an input layer, one or more hidden layers and an output layer [19]. The input layer usually merely passes data along without modifying it. Most of the computation happens in the hidden layers. The output layer converts the hidden layer activation to an output, such as a classification. The outputs of each hidden layer serve as the inputs for the next hidden layer. The number of neurons in the output layer is equal to the number of classes trained for the neural network [33].

3.3 Computer Vision

Computer vision is the field of computer science that focuses on replicating parts of the complexity of the human vision system and enabling computers to identify and process objects in images and videos in the same way that humans do. Until recently, computer vision only worked in limited capacity.

It is the area of study in which computers are empowered to visualize, recognize and process what they see in a similar way as that of humans [24]. The main aim of computer vision is to generate relevant information from image and video data in order to deduce something about the world [25] [26]. It can be classified as a sub-field of artificial intelligence and machine learning. This is quite different from image processing, which involves manipulating or enhancing visual information and is not concerned about the contents of the image. Applications of computer vision include image classification, visual detection, 3D scene reconstruction from 2D images, image retrieval, augmented reality, machine vision and traffic automation [27] [28] [29] [30] [31] [32].

3.3.1 Evolution of Computer Vision

The tasks that computer vision could perform were very limited and required a lot of manual coding and effort by developers and human operators, before the advent of deep learning

[34]. For instance, if we wanted to perform facial recognition, you would have to perform the following steps:

1. ***Create Database***: We had to capture individual images of all the subjects you wanted to track in a specific format.
2. ***Annotate Images*:** Then for every individual image, we would have to enter several key data points, such as distance between the eyes, the width of nose bridge, distance between upper-lip and nose, and dozens of other measurements that define the unique characteristics of each person.
3. ***Capture new images***: Next, we would have to capture new images, whether from photographs or video content. And then you had to go through the measurement process again, marking the key points on the image. You also had to factor in the angle the image was taken.

After all this manual work, the application would finally be able to compare the measurements in the new image with the ones stored in its database and tell you whether it corresponded with any of the profiles it was tracking. In fact, there was very little automation involved and most of the work was being done manually. And the error margin was still large. A different approach [35] provided by machine learning to solving computer vision problems. With machine learning, developers no longer needed to manually code every single rule into their vision applications. Instead they programmed "features," smaller applications that could detect specific patterns in images. They then used a statistical learning algorithm such as linear regression, logistic regression, decision trees or support vector machines (SVM) to detect patterns and classify images and detect objects in them.

Today, machine learning is a necessary component of many computer vision algorithms [33]. These algorithms are typically a combination of image processing and machine learning techniques. The major requirement of these algorithms is to handle large amounts of image/video data and to be able to perform computation in real-time for wide range of applications. For example, real-time detection and tracking.

Machine learning helped solve many problems that were historically challenging for classical software development tools and approaches. For instance, years ago, machine learning engineers were able to create a software that could predict breast cancer survival

windows better than human experts. However, building the features of the software required the efforts of dozens of engineers and breast cancer experts and took a lot of time develop.

Deep learning provides a fundamentally different approach to doing machine learning. Deep learning relies on neural networks, a general-purpose function that can solve any problem representable through examples. When we provide a neural network with many labeled examples of a specific kind of data, it will be able to extract common patterns between those examples and transform it into a mathematical equation that will help classify future pieces of information. For example, creating a facial recognition application with deep learning only requires us to develop or choose a preconstructed algorithm and train it with examples of the faces of the people it must detect. Given enough examples (lots of examples), the neural network will be able to detect faces without further instructions on features or measurements [36].

3.3.2 Working Process for Computer Vision

Modern computer vision systems combine image processing with machine learning and deep learning techniques. Hence, developers combine different software (i.e. OpenCV) and AI algorithms to create a multi-step process, a computer vision pipeline [37].The organization and setup of a computer vision system vary based on the application and use case. However, all computer vision systems contain the same typical functions:

Step 1: Image acquisition: The digital image of a camera or image sensor provides the image data or video. Technically, any 2D or 3D camera or sensor can be used to provide image frames.

Step 2: Pre-processing: The raw image input of cameras needs to be preprocessed to optimize the performance of the subsequent computer vision tasks. Pre-processing includes noise reduction, contrast enhancement, re-scaling, or image cropping.

- ***Image Resize:*** The input images are resized according to input image layer size of pre-trained model. As the input image layer sizes of pre-trained is of resolution. So, all the training and testing images are resized according to the image resolution [4].

- ***Image Filter and Enhancement:*** The information obtained directly from the camera contains a lot of noise. This section uses a Gaussian filtering method to reduce the impact of these noise disturbances. Then a local contrast enhancement algorithm is used to effectively improve all or part of the features of the image [3].
- ***Edge Detection:*** To implement the canny edge detection method, the 3x3 Sobel operator is used to calculate the gradient value of the input image [5]. The magnitude of the gradient will be calculated according to the formula and the direction of the edge and determined as an equation. Then it will suppress any pixel values that are not considered edges, and finally apply the double threshold method to determine potential edges [3][4].

Step 3: Computer vision algorithm: The image processing algorithm, most popularly a deep learning model, performs object detection, image segmentation, and classification on every image or video frame [37][24].

Step 4: Automation logic: The AI algorithm output information needs to be processed with conditional rules based on the use case. This part performs automation based on information gained from the computer vision task. For example, pass or fail for automatic inspection applications, match or no-match in recognition systems, flag for human review in insurance, security, military, or medical recognition applications [38].

3.4 Convolutional Neural Networks

The Convolutional Neural Networks are specialized type of neural network model designed for working with two-dimensional image data although they can be used with one-dimensional and three-dimensional data. Central to the convolutional neural network is the convolutional layer that gives the network its name. This layer performs an operation called a ‘convolution’. In the context of a convolutional neural network, a convolutional is a linear operation that involves the multiplication of a set of weights with the input, much like a traditional neural network. Given that the technique was designed for two-dimensional input, the multiplication is performed between an array of input data and a two-dimensional array of weights, called a filter or a kernel.

The filter is smaller than the input data and the type of multiplication applied between a filter-sized patch of the input and the filter is a dot product. A dot product is the element-wise multiplication between the filter-size patch of the input and filter, which is then summed, always resulting in a single value. Because it results in a single value, the operation is often referred to as the 'scalar product'. Using a filter smaller than the input is intentional as it allows the same filter smaller than the input is intentional as it allows the same filter (set of weights) to be multiplied by the input array multiple times at different points on the input. Specifically, the filter is applied systematically to each overlapping part or filter-sized patch of the input data, left to right, top to bottom.

This systematic application of the same filter across an image is good idea. If the filter is designed to detect a specific type of feature in the input, then the application of that filter systematically across the entire input image that allows the filter to convolve it with some number of filters of size K*K, and take the stride as 2 or 3 and with or without padding. This will give us an output of 3-dimensional image. We then convolve this output further and get an output of K*L*M. So many hyperparameters that we can manipulate while building a convolutional network. These include the number of filters, size of filters, stride to be used, padding.

In a convolutional network, there are mainly three types of layers:

- Convolutional Layer
- Pooling Layer
- Fully Connected Layer

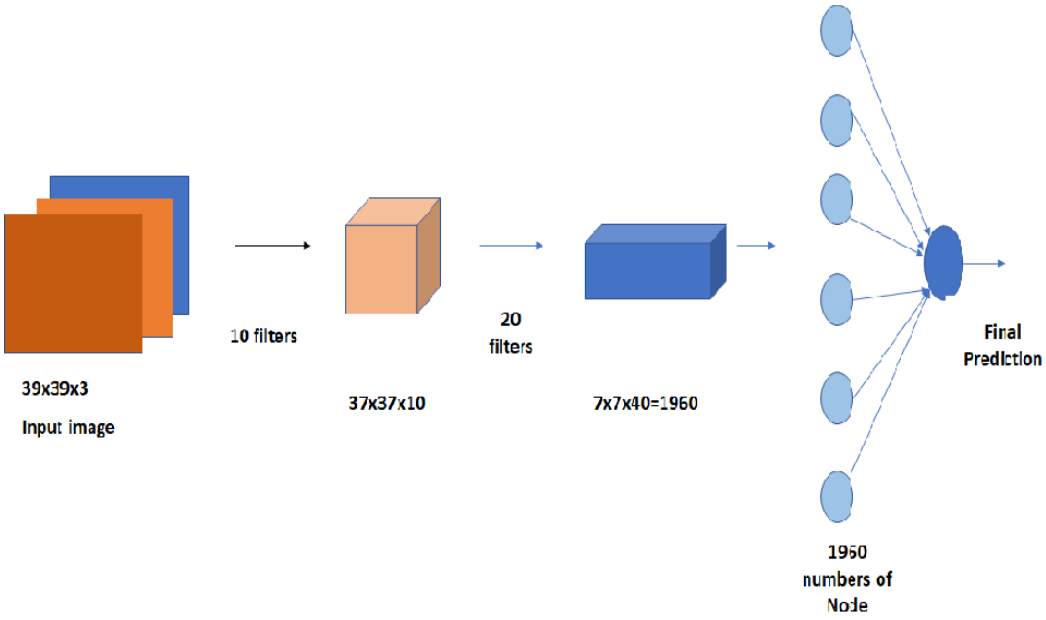

Figure 3.1: How images are filter using convolutional layers and produce final result

There are various types of artificial neural networks that are considered to be very important such as Radial basis function neural network, Feed-forward neural network, Convolutional neural network, Recurrent neural network, Modular neural network etc. Among these types of networks, the convolutional neural networks (CNNs) are effective in applications such as image/video recognition [34], semantic parsing, natural language processing and paraphrase detection [35]. A convolutional neural network typically comprises of three layers – Convolutional layer, pooling layer and Fully-connected layer.

3.4.1 Convolutional Layer

A convolutional neural network consists of one or more convolutional layers. These layers can either be pooled or fully connected [35]. A convolutional layer generally executes tasks that require heavy computation. It comprises of a set of filters that have the ability to learn. Though the filters are small in size, they reach to the entire depth of the input. The dimensions of a filter are generally represented by $h * w * d$, where 'h' denotes the height of the length of the filter, 'w' denotes the width while 'd' denotes the depth of the feature filter which is equal to the number of color channels present. In general, the convolution process is

executed by a feature filter upon sliding on the input layer of the neural network, as a result of which a feature map is generated. The layer executing the convolution process is known as a convolutional layer. Hence, the networks that consist of convolutional layers are called as convolutional neural networks. As shown in the figure bellow, in the initial stages, the input layer is searched for any specific pattern by the filter. During the training of the algorithm, the filter searches for the sake of learning to recognize a pattern which eventually becomes a search to validate the existence of a specific pattern, during the testing stages. In reality, many feature filters exist, learning to recognize various patterns[39].

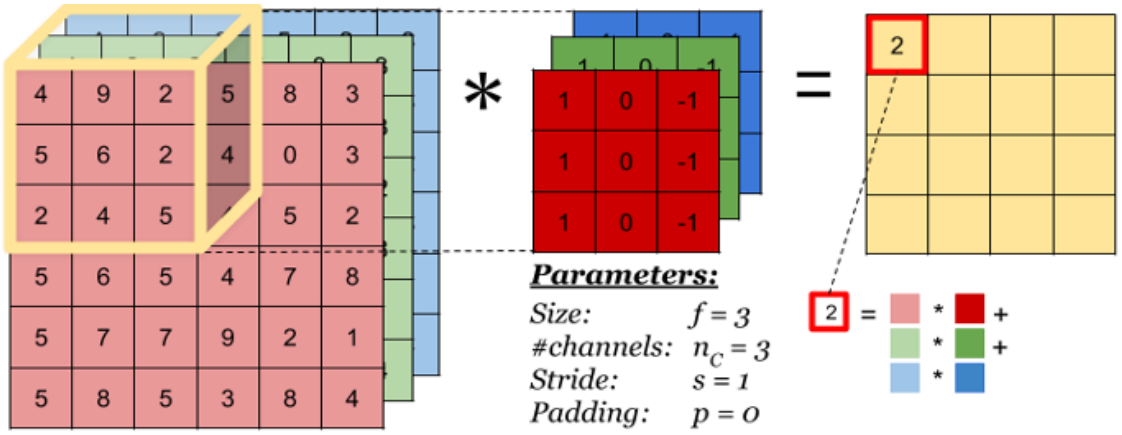

Fig 3.2: Feature extraction using different filter on input image matrix (Source: https://medium.com/)

3.4.2 Pooling Layer

Convolutional layers are the basic building blocks of a convolutional neural network used for computer vision applications such as image recognition. A convolutional layer slides a filter over the image and extracts features resulting in a feature map that can be fed to the next convolutional layer to extract higher-level features. Thus, stacking multiple convolutional layers allows CNNs to recognize increasingly complex structures and objects in an image[40].

A major problem with convolutional layers is that the feature map produced by the filter is location-dependent. This means that during training, convolutional neural networks

learn to associate the presence of a certain feature with a specific location in the input image. This can severely depress performance. Instead, we want the feature map and the network to be translation invariant (an expression that means that the location of the feature should not matter). The basic procedure of pooling is very similar to the convolution operation. We select a filter and slide it over the output feature map of the preceding convolutional layer. The most commonly used filter size is 2×2 and it is slid over the input using a stride of 2. Based on the type of pooling operation we selected, the pooling filter calculates an output on the receptive field [41]. The pooling layers used are briefly described as follows:

1. ***Max Pooling:*** Here, the filter simply selects the maximum pixel value in the receptive field. For example, if we have 4 pixels in the field with values 3, 9, 0, and 6, we select 9.

2. ***Average Pooling: This*** works by calculating the average value of the pixel values in the receptive field. Given 4 pixels with the values 3, 9, 0, and 6, the average pooling layer would produce an output of 4.5. Rounding to full numbers gives us 5.

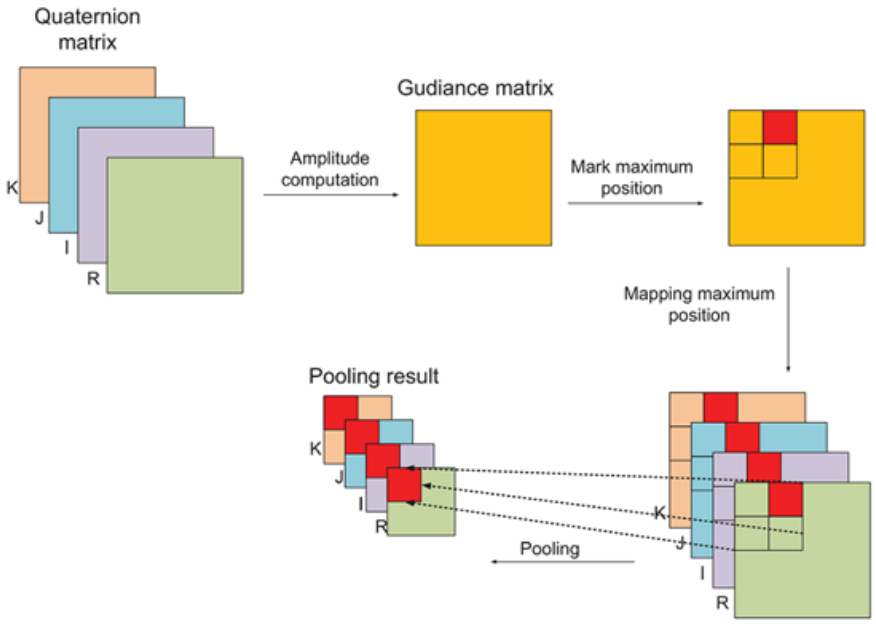

Figure 3.3: Average pooling and Max pooling done over the matrices of an image. (Source: https://discuss.pytorch.org/)

3.5 EAST: An Efficient and Accurate Scene Text Detector

The key component of the EAST algorithm is a neural network model, which is trained to directly predict the existence of text instances and their geometries from full images. The model is a fully-convolutional neural network adapted for text detection that outputs dense per-pixel predictions of words or text lines. This eliminates intermediate steps such as candidate proposal, text region formation and word partition. The post-processing steps only include thresholding and NMS on predicted geometric shapes. The detector is named as EAST, since it is an Efficient and Accuracy Scene Text detection pipeline.

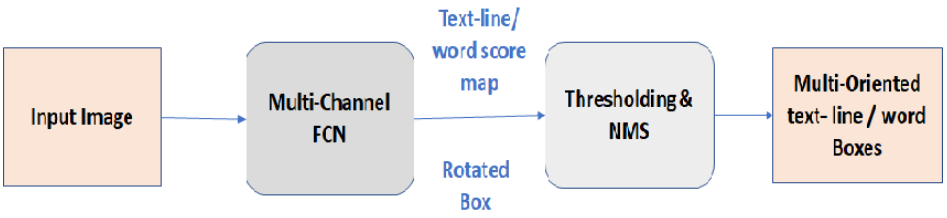

Figure 3.4: Working Pipeline of an Efficient and Accurate Scene Text Detector

3.5.1 Pipeline

A high-level overview of our pipeline is illustrated in the below diagram. The algorithm follows the general design of Dense-Box [9], in which an image is fed into the FCN and multiple channels of pixel-level text score map and geometry are generated. One of the predicted channels is a score map whose pixel values are in the range of [0, 1]. The remaining channels represent geometries that encloses the word from the view of each pixel. The score stands for the confidence of the geometry shape predicted at the same location. We have experimented with two geometry shapes for text regions, rotated box (RBOX) and quadrangle (QUAD), and designed different loss functions for each geometry. Thresholding is then applied to each predicted region, where the geometries whose scores are over the predefined threshold is considered valid and saved for later non-maximum-suppression. Results after NMS are considered the final output of the pipeline.

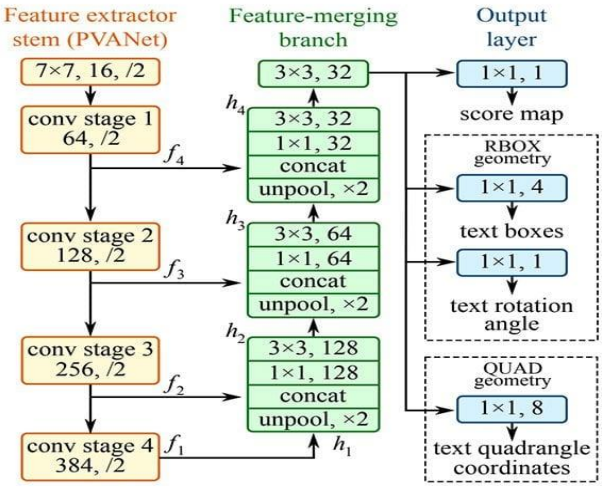

Figure 3.5: Architecture of EAST a text detection model

3.5.2 Loss Functions

The loss can be formulated as:

$$L = L_s + \lambda(g) \times L(g) \tag{1.2}$$

Where Ls and Lg represents the losses for the score map and the geometry, respectively, and λ (g) weighs the importance between two losses. In our experiment, we set λ(g) to 1.

3.5.3 Results

The input image is passed through the pipeline of EAST model and the results are shown in Figure 3.6. We all are well aware of the class imbalance problem present in object detection datasets. The number of samples for background class is generally very high in number and now that we are treating every 1x1 box (basically every pixel) as output, the number of background samples becomes huge. In order to tackle this problem of class imbalance, EAST uses a modified version of cross entropy called **Balanced/Weighted Cross Entropy.**

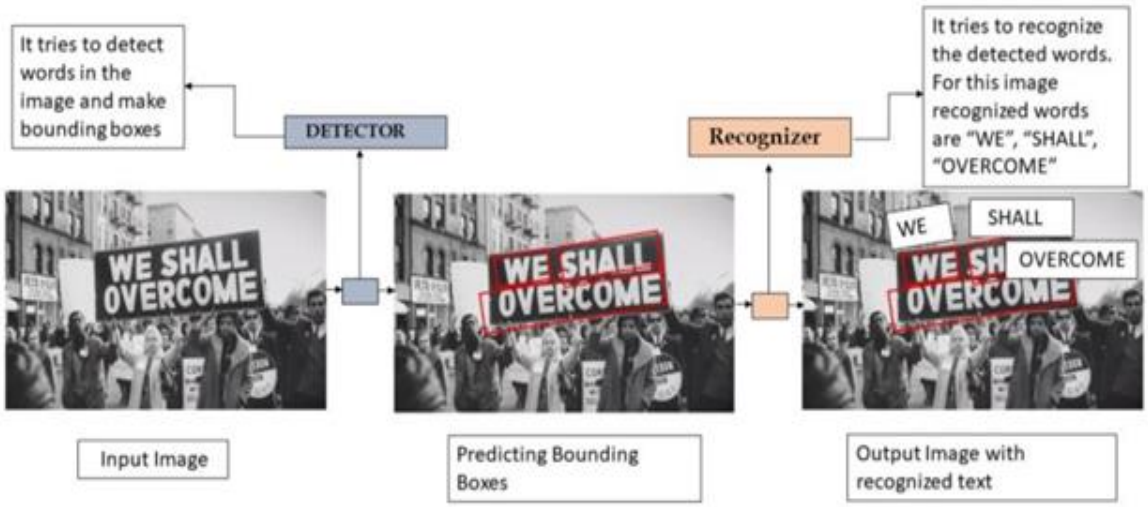

Figure 3.6: Predicted Bounding Boxes for a Blurry Input Image

Figure 3.7: Few results of East Text detection

3.6 YOLO Version 3

YOLO stands for You Only Look Once. It's an object detector that uses features learned by a deep convolutional neural network to detect an object. The state-of-the-art object detector YOLOv3 is designed to achieve high accuracy along with real-time performance. YOLOv3 is an improvement over the previous version of YOLO. It uses a single neural network, which predicts the objects position and class score in a single iteration. This is achieved by considering object detection problem as a regression problem, which in turn changes the input images to their corresponding class probabilities and positions. YOLO generates Many S x S grids from the input image and boundary boxes B are predicted, which consists of height, width, box center x and y. Each of these boxes have their own P (object probability) value and predicts the number of classes in it as C and has a conditional class probability P class in the S x S having an object in it. The overall prediction of the network is S x S x (Bx5 + C) in which the digit 5 represents each box coordinates as 4 and 1 as object probability.

During the test, the network computes the number of classes present in each grid by using the bellow equation. P-min is defined at the start of the test and system detects only the objects whose P-class > P-min. During the post-processing stage, the duplicated detection of the same object is omitted using Non-maximal suppression.

$$\mathrm{P}\,(\mathrm{class}-\mathrm{I}) \;=\; \mathrm{P}(\mathrm{class}-\mathrm{I}\mid \mathrm{object}) \;*\; \mathrm{P}(\mathrm{object}) \qquad (1.3)$$

Here, P(class-I) is the probability of 1th class. P(Object) is the probability of grid containing the object and P(class-I | object) is the conditional class probability of the 1th class in which the object is present.

In YOLO, only the bounding boxes with the greatest value of confidence are selected since every grid-cell is predicting multiple bounding boxes. Therefore, YOLO generates a tensor as an output whose value is equal to S x S x (B $*$ 5 + C) [8]. In YOLOv3, the bounding boxes have been replaced by 'Anchors' which resolve the unstable gradient issue that used to occur while training of the algorithm. Therefore, YOLOv3 predicts outputs with confidence scores by generating a vector of bounding boxes whenever an input is given to the algorithm in the form of an image or a video.

3.6.1 Bounding Box Prediction

The proposed system predicts bounding boxes using dimension clusters as anchor boxes [15]. The network predicts 4 coordinates for each bounding box Tx, Ty, Tw, Th. If the cell is offset from the top left corner of the image By (cx, cy) and the bounding box prior has width and height Pw, Ph, then the predictions correspond to:

$$B(x) = \sigma(T_x) + c_x \quad (1.4)$$

$$B(y) = \sigma(T_y) + c_y \quad (1.5)$$

$$B(w) = P_w e^{T_w} \quad (1.6)$$

$$B(h) = P_h e^{h} \quad (1.7)$$

During training we use sum of squared error loss. If the ground truth for some coordinate prediction is G(d) our gradient is the ground truth value (computed from the ground truth box) minus our prediction: G(d) – P(d). This ground truth value can be easily computed by inverting the equations above. YOLOv3 predicts an object-ness score for each bounding box using logistic regression. This should be 1 if the bounding box prior overlaps a ground truth object by more than any other bounding box prior. If the bounding box prior is not the best but does overlap a ground truth object by more than some threshold we ignore the prediction, following [17]. We use the threshold of 0.5. Unlike [17] our system only assigns one bounding box prior for each ground truth object. If a bounding box prior is not assigned to a ground truth object it incurs no loss for coordinate or class predictions, only object-ness.

3.6.2 Class Prediction

Each box predicts the classes the bounding box may contain using multilabel classification. We do not use a soft-max as we have found it is unnecessary for good performance, instead we simply use independent logistic classifiers. During training we use binary cross-entropy loss for the class predictions. This formulation helps when we move to more complex domains like the **ICDAR MLT 2019** [7]. In this dataset there are many overlapping labels (i.e. Woman and Person). Using a soft-max imposes the assumption that each box has

exactly one class which is often not the case. A multilabel approach better models the data[27].

3.6.3 Feature Extractor

A new network is used for performing feature extraction. Our new network is a hybrid approach between the network used in **YOLOv3, Darknet-19**, and that make residual network stuff. Our network uses successive 3 × 3 and 1 × 1 **convolutional layers** but now has some shortcut connections as well and is significantly larger. It has 53 convolutional layers so we call it Darknet-53[27].

Each network is trained with identical settings and tested with images of size 256×256, single crop accuracy. Run times are measured on windows 10 operating system. Thus Darknet-53 performs with state-of-the-art classifiers but with fewer float point operations and less speed. Darknet-53 is better than ResNet-101 and 1.5× faster. Darknet-53 has similar performance to ResNet-152 and is 2× faster. Darknet-53 also achieves the highest measured float point operations per second. This means the network structure better utilizes the GPU, making it more efficient to evaluate and thus faster. That is mostly because Res-Nets have just way too many layers and are not very efficient.

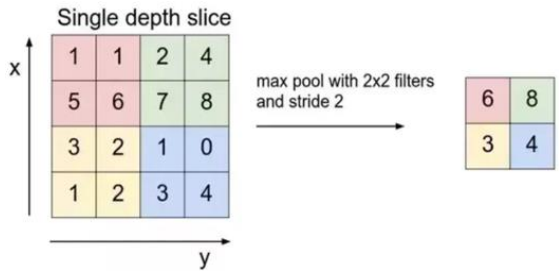

Figure 3.8: Features are extracted using Pooling method

3.6.4 Anchor Box

YOLO can work well for multiple objects where each object is associated with one grid cell. But in the case of overlap, in which one grid cell actually contains the center points of two different objects, we can use something called anchor boxes to allow grid cell to detect multiple objects.

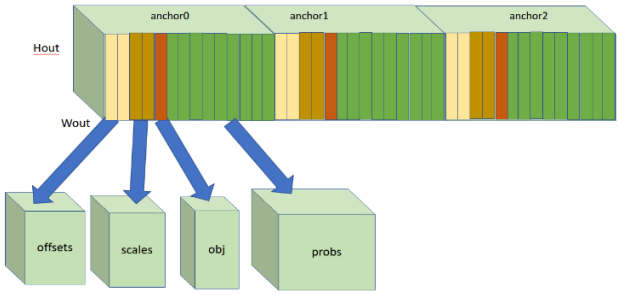

Figure 3.9: Anchor Boxes are predicted during testing phase.

The test image is first broken up into a grid and the network then produces output vectors, one for each grid cell. These vectors tell us if a cell has an object in it, what class the object is, and the bounding boxes for the object. Since we're using two anchor boxes, we'll get two predicted anchor boxes for each grid cell. Some, in fact most of the predicted anchor boxes will have a very low PC value. After producing these output vectors, we use non-maximal suppression to get of unlikely bounding boxes. For each class, non-maximal suppression gets rid of the bounding boxes that have a PC lower than some given threshold.

3.6.5 Non-Maximal Suppression (NMS)

YOLO uses Non-Maximal Suppression (NMS) to only keep the best bounding box. The first step in NMS is to remove all the predicted bounding boxes that have a detection probability that is less than a given NMS threshold. We set this NMS threshold to 0.7 for text detection and identification. This means that all predicted bounding boxes that have a detection probability less than 0.7 will be removed.

3.6.6 Intersection Over Union Threshold (IOU)

After removing all the predicted bounding boxes that have a low detection probability, the second step in NMS, is to select the bounding boxes with the highest detection probability and eliminate all the bounding boxes which Intersection over Union (IOU) value is higher than a given IOU threshold. In our code, we set this IOU threshold to 0.4. This means that all predicted bounding boxes that have an IOU value greater than 0.4 with respect to the best bounding boxes will be removed.

3.6.7 Architecture

YOLOv2 used a feature extractor known as the Darknet-19, which consisted of 19 convolutional layers. The newer version of this algorithm, YOLOv3 uses a new feature extractor known as Darknet-53 which, as the name suggests, uses 53 convolutional layers while the overall algorithm consists of 75 convolutional layers and 31 other layers making it a total of 106 layers [36]. Pooling layers have been removed from the architecture and replaced by another convolutional layer with stride '2', for the purpose of down-sampling. This key change has been made to prevent the loss of features during the process of pooling.

The newer architecture boasts of residual skip connections, and up-sampling. **The most salient feature of v3 is that it makes detections at three different scales.** YOLO is a fully convolutional network and its eventual output is generated by applying a 1 x 1 kernel on a feature map. In YOLO v3, **the detection is done by applying 1 x 1 detection kernels on feature maps of three different sizes at three different places in the network.**

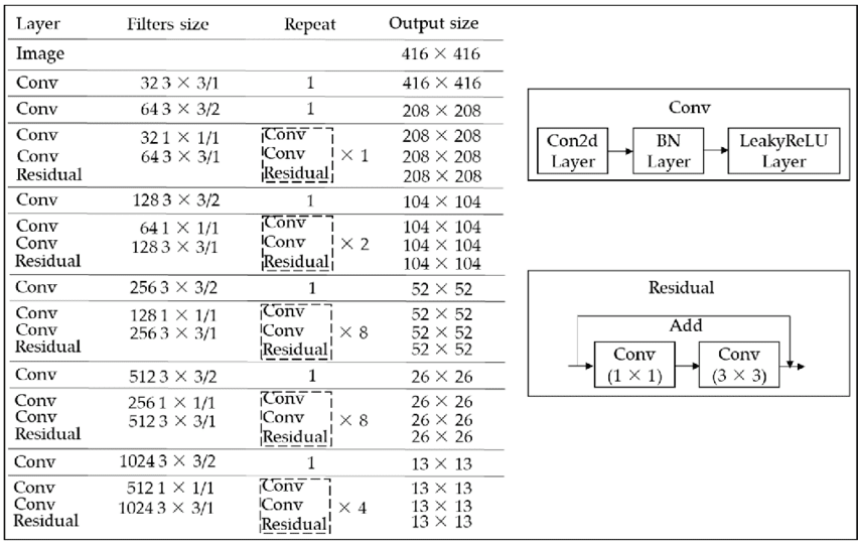

Figure 3.10: Architectural Diagram of YOLO version 3

In the Figure 2.8, the size of the input image is 416 x 416. As mentioned in the earlier section, the total number of layers in YOLOv3 is 106. As shown in the network

architecture diagram Figure 2.8, the input image is down sampled by the network for the first 81 layers.

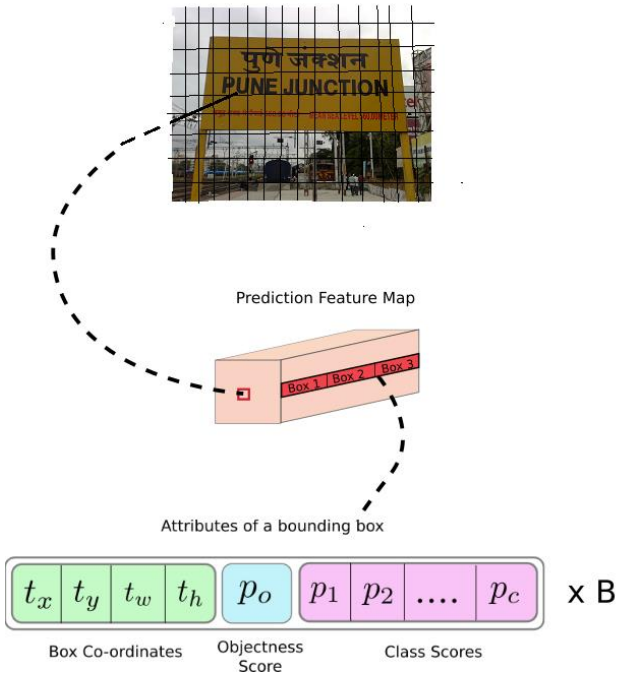

Figure 3.11: Each Grid box are analyzed in the image

Since the 81st layer has a stride of 32, the 82nd layer performs the first detection with a feature map of size 13 x 13. Since a 1 x 1 kernel is used to perform the detection, the size of the resulting detection feature map is 13 x 13 x 255 which is responsible for the detection of objects at scale 3. Following this, the feature map from 79th layer is up sampled by 2x after subjecting it to a few convolutional layers, resulting in the dimensions 26 x 26. This is then concatenated with the feature map from 61st layer. The features are fused by subjecting the concatenated feature map to a few more 1 x 1 convolutional layers. As a result, the 94th layer performs the second detection with a feature map of 26 x 26 x 255, which is responsible for the detection of objects at scale 2. Following the second detection, the feature map from 91st layer is up sampled by 2x after subjecting it to a few convolutional layers, resulting in the dimensions 52 x 52. This is then concatenated with the feature map from 36th layer. The features are fused by subjecting the concatenated feature map to a few more 1 x 1

convolutional layers. As a result, the 106th layer performs the third and final detection with a feature map of 52 x 52 x 255, which is responsible for the detection of objects at scale.

Figure 3.12: Detected text regions in sample images.

Chapter 4

Experiment & Results

In this chapter, all the related experiment and its results are given. We will see how the datasets are collected and annotated in the YOLO format based. At first the experiment was done using 70 to 80 Natural scene images, then the entire datasets are applied and trained this yolo model to make desired output. Initially, we were getting the detected results in overlapped fashion, because of that we got the recognized text with more numbers of error. Finally, after tuning few parameters of the model and trained over 400 numbers of NSI and 2700 numbers of ICDAR images, we were able to get much better results which are discussed below. Instead of previous chapter, we also discussed little bit about Tesseract OCR in this chapter to attach some direct resultant images.

4.1 Datasets Preparation

The YOLO V3 model itself are trained on NSI data set. But as the ground-truth value are not in desired form so we needed to annotates the text information of the original data images and makes a database in the format of VOC2007 data set. The steps are as follows:

1. Rename the original data images and like NSI_1.jpg, NSI_2.jpg so on.
2. Initially, Images text bounding boxes are annotated with Label-IMG tool, and corresponding XML bounding information files are generated at the same time. They are similar to SVT data set, only corresponding to a single image with VOC format.

3. The image labeling XML file is parsed to generate the corresponding TXT format label file.
4. Generates train.txt and Val.txt for specifying training and test images. The first 400 JPG images were selected for training, and the 100 images were verified.

4.1.1 Annotation Tool

An annotation tool for creating ground-truth for around 80 images for initial training of model to check whether our trained model really works well or not.

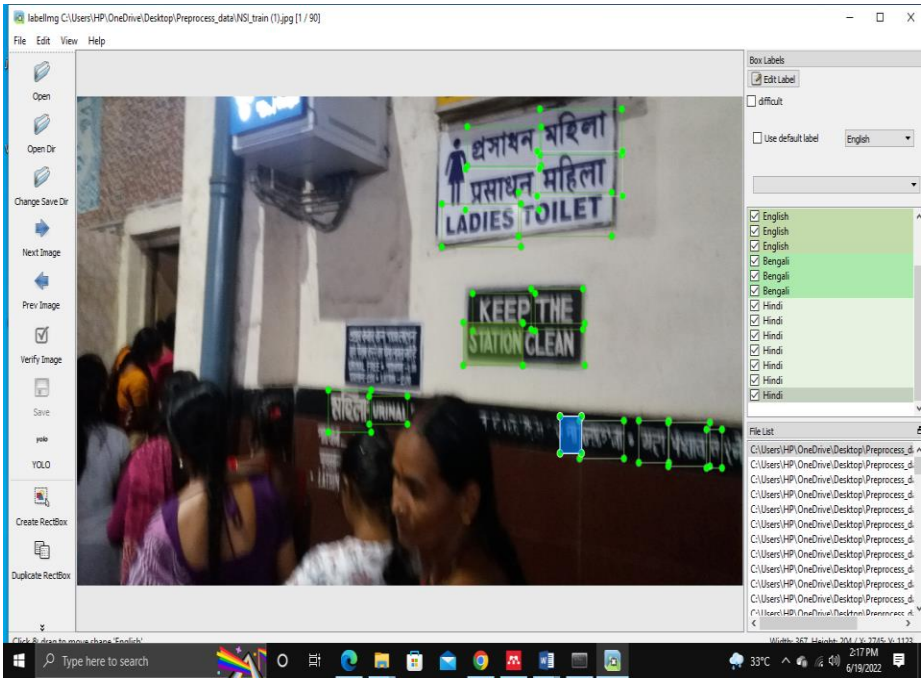

Figure 4.1: Label-IMG tool for Labelling text images for creating ground-truth

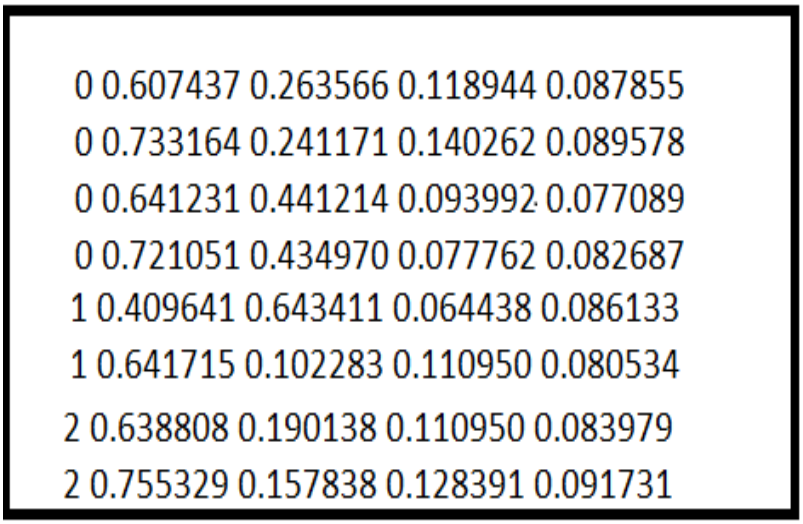

Figure 4.2: Ground-truth (English, Bengali, Hindi) Made by Label-IMG Tool

4.1.2 Conversion of Ground-Truth Format

As time did not permit us to make ground- truth for each and every image using tool. To compensate this issue, we got solution by writing mathematical formulation for converting the ground-truth format [42].

For example, this is a ground-truth values for an image consist of English, Bengali, Hindi texts. But it is not the right format which is needed to our proposed method.

2427.5	183.5	408	120	b
2877.5	57.5	396	174	b
2415.5	345.5	426	168	h
2871.5	267.5	414	192	h
2481.5	957.5	336	138	e
2275	535	460	124	e
2763.5	477.5	498	114	e
2835.5	939.5	252	138	e
2415.5	1107.5	348	144	e
2781.5	1101.5	312	138	e
1569.5	1407.5	234	138	b
1815.5	1443.5	222	108	e

Figure 4.3: Ground Truth Before Conversion

In this figure, the coordinate format of the label file is: Label, X, Y, W, h. Label, namely "0", X and Y, are the central coordinates of bounding Box, namely (0.607437,0.263566), W and H are the ratio of width and height of Box and image, namely 0.118944 and 0.087855. This output is same as earlier which we got using tool. This proves that our mathematical code not only works properly but also convert entire dataset into desired form in very less time.

```
def convert_label(x,y,w,h):
    img = cv2.imread(r'C:\Users\HP\Downloads\NSI_train_2.jpg')
    h1,w1=img.shape[:2]
    dw=1./w1
    dh=1./h1
    x1=(x+w/2)
    y1=(y+h/2)
    x1=x1*dw
    w=w*dw
    y1=y1*dh
    h=h*dh
    return (x1,y1,w,h)
```

Figure 4.4: Mathematical Code for Converting Ground-truth

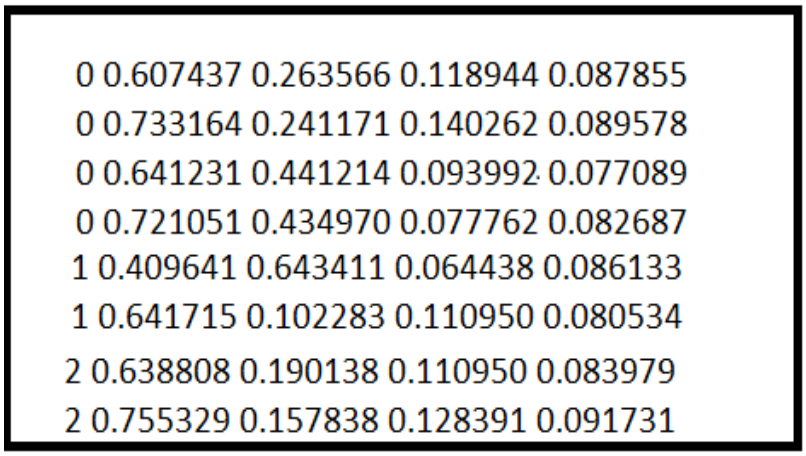

Figure 4.5: Ground Truth After Conversion

4.2 Training Process

YOLOv3 model has its own unique open source framework Darknet, which can be run on BOTH CPU and GPU. In this thesis work, CUDA7.5 and CUDNN provided by Google Co-lab compatible with YOLO version are configured and run on GPU for experimental training.

4.2.1. Experimental Environment

Hardware environment: Intel(R)-Core (TM) I3-5790 2.70GHz CPU

Software environment: Windows 10, Google Co-lab, openCV3.4.20

4.2.2 Selection parameters

In this work, the experimental training strategy by adopting the technology of fine - tuning, using ImageNet pre-training weights files before 23 layers weight training, some parameters such as decay at parameter is set to the initial parameters of the original network ImageNet, only need to modify learning rate(vector), max batches(iterations), classes (number of categories in model checking), filters (the last layer convolution kernel number) four parameters. In the experiment in this work, max batches = 60000, classes = 3, Filters = num*(classes+coords+1) = 30. The main parameter is selected as learning rate. In this paper, NSI and ICDAR [5] data set and homemade database were used for training test, mainly for NSI database. Therefore, the selection of parameters mainly considers the training based on

NSI database. During the model training, different learning rates will have different influences on the training result loss function Loss. In this experiment, Python was used to visualize avg-loss. In the Figures, for example, in batches of AVG-loss of model training under different Learning rates 60000 times, Table below shows the loss function loss value of loss under different Learning rates at the end of training.

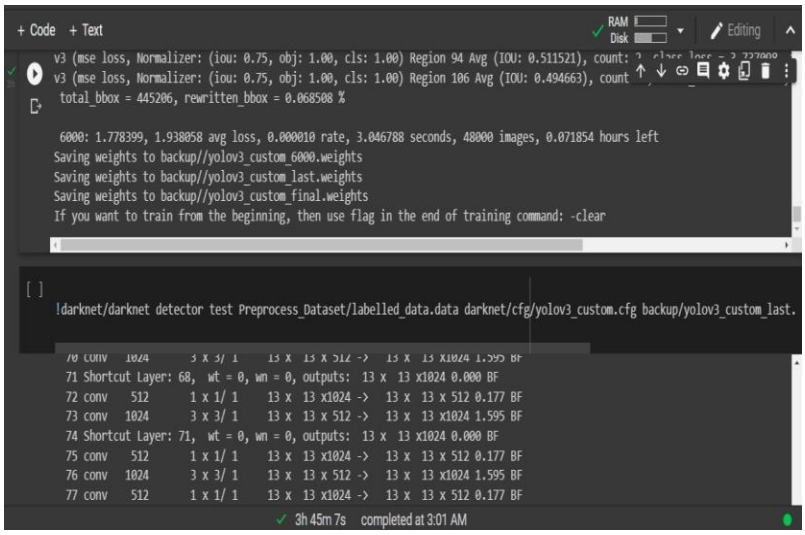

Figure 4.6: During Training Process

Other parameters were set during the training time,

classes=3

train=train.txt

valid=test.txt

names=Label.Names

backup = backup/

- **classes:** Number of class in our data set
- **train:** Train file path
- **test:** Test file path
- **names:** Class name file
- **backup:** Where we want to store the yolo weights file

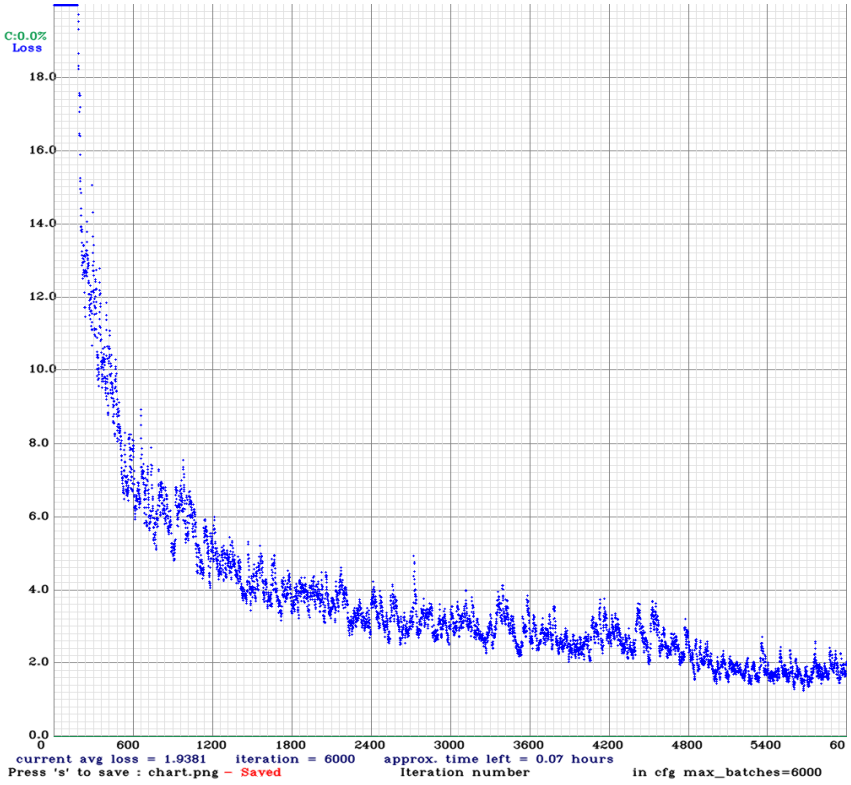

Figure 4.7: Visualization curve of AVG-loss when LR =0.001 of NSI Dataset

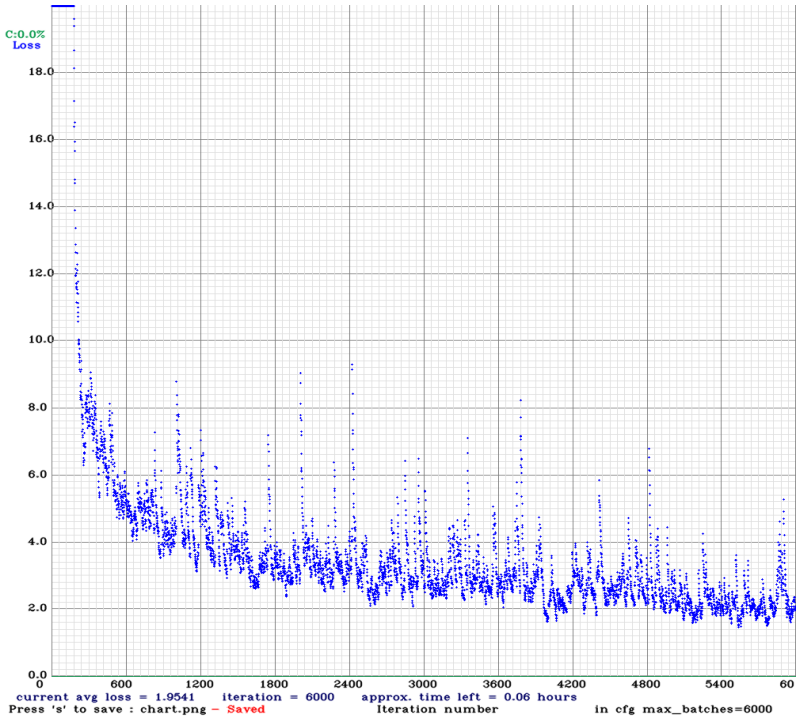

Figure 4.8: Visualization curve of AVG-loss when LR =0.001 of ICDAR MLT 2019 Dataset

Table 1: Loss Details

Learning Rate	0.001	0.0001
Loss	1.9541	0.8540

From the visualization graph of AVG-loss, it can be intuitively seen that as the number of iterations increases, the AVG-loss trained in the model keeps decreasing, and the number of iterations reaches about 1000. The reduction range of AVG-loss obviously slows down, showing a convergence trend. However, when learning-Rate =0.001, the AVG-loss curve diverged several times in the later stage of training, compared with the stable convergence when learning-rate=0.0001. By comparing the iteration ending loss in Table above, then the experiment was conducted with learning-rate=0.0001 but the graph was not shown.

4.3 Experiment Results

To understand the performance of any system for a given dataset, certain evaluation criteria need to be ascertained. These criteria considered here include the choice of some standard metrics and protocols like Precision, Recall, F-measure and Accuracy that reflect the quality of performance of any method concerned. These metrics are defined by the following equations:

$$P = \frac{TP}{TP+FP} \tag{1}$$

$$R = \frac{TP}{TP+FN} \tag{2}$$

$$F - measure = \frac{2\times P\times R}{P+R} \tag{3}$$

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \tag{4}$$

Where, P and R signify Precision and Recall in Eqs. (1) and (2) respectively. Also, TP denotes True Positives for pixels within the ground truth text boxes. The pixels falling outside the ground truth text boxes are considered as True Negatives denoted as TN. Denoted

as FP, False Positives signify pixels wrongly identified as part of the ground truth text boxes and False Negatives, denoted as FN, signify those ground truth text box pixels that are falsely excluded.

Initially with custom made set of images from randomly chosen NSI data-set model was giving very good result as depicted in Figure 4.9.

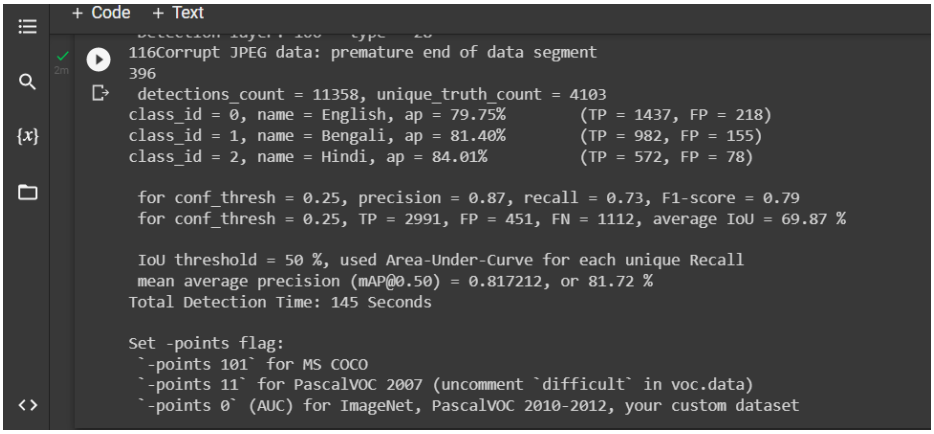

Figure 4.9: Test result of few images to check how much precision, recall, f1 score is

Table 2: Text detection results on the combined pool of NSI and BDI from MITDI dataset.

The Dataset	**Precision**	**Recall**	**IOU**	**Accuracy**
NSI+BDI	**67.78**	**71.45**	**68.49**	**70.02**

Here accuracy has not been up to the mark for entire dataset may be due overfitting problem. To make more accurate our model, we need to tune some parameter like learning rate from 0.001 to 0.0001. Then it is expected that model will give far more accuracy than now. But due to time constrain, we could not make it possible now.

4.4 Tesseract OCR

Tesseract is an open source text recognition (OCR) Engine, available under the Apache 2.0 license. It can be used directly using an API to extract printed text from images. It supports a wide variety of languages. Tesseract doesn't have a built-in GUI. Tesseract is compatible with

many programming languages and frameworks through wrappers. It can be used with the existing layout analysis to recognize text within a large document, or it can be used in conjunction with an external text detector to recognize text from an image of a single text line.

To recognize an image containing a single character, we typically use a Convolutional Neural Network (CNN). But, text of arbitrary length is a sequence of characters, and such problems are solved using RNNs and LSTM is a popular form of RNN which is implemented inside Tesseract OCR.

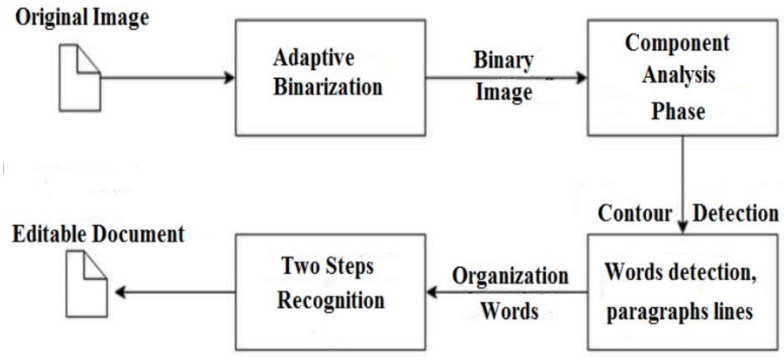

Figure 4.10: Tesseract OCR working model Diagram

4.5 Pipeline

The input image is given to the below Pipeline, and the desired output is generated.

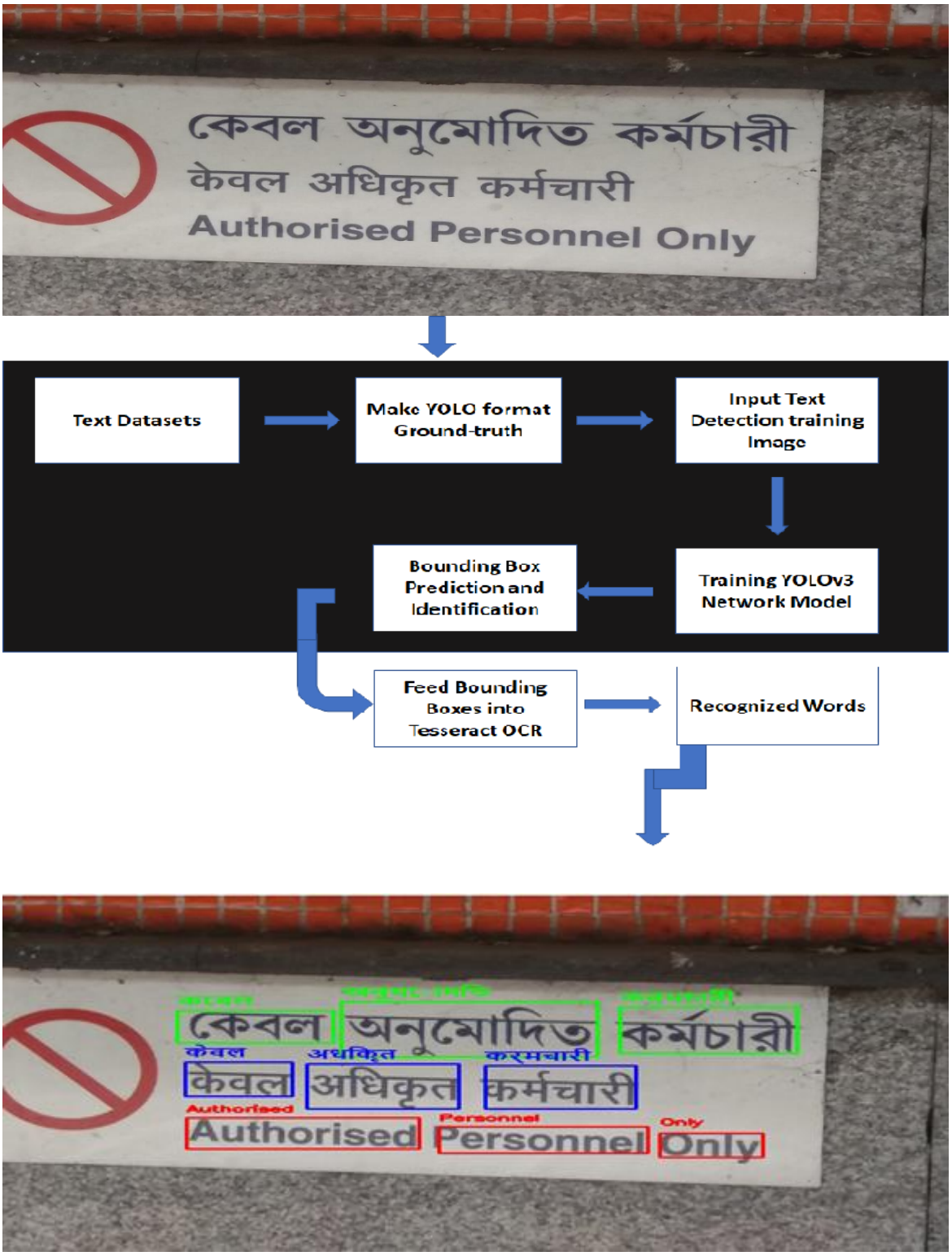

Figure 4.11: An overall mechanism of the proposed system. Here different colors are used for different language for easy identification of which language the texts are identified.

Few correctly predicted results during testing on natural scene images are shown in Figure 4.12.

Figure 4.12: Samples of outcome obtained from the proposed system.

The proposed framework is able to detect and recognize most of the texts in various images. However, in certain cases, it is faced with some limitations with respect to detection and OCR process. Sample of these issues are depicted in Figure 4.13.

Figure 4.13. Sample images showing the performance drawbacks.

Chapter 5

Conclusion

This thesis presents a YOLO v3 based end-to-end framework for real-time and static natural scene text detection and recognition and evaluates the performance of these algorithms on the detection and recognition using Tesseract-OCR tool.

Experiments have been carried out to evaluate the classification performance of this deep-learning algorithm. After the preparation of dataset, the algorithm has been trained on the train-dataset. The trained model has been tested on the test images, from which the number of true positives, true negatives, false positives and false negatives have been identified for each images of the detections made by the YOLOv3 deep-learning model on the test images. Using these results the Accuracy, Precision, Recall and finally the F1 score of the models have been calculated. The results of the experiment have been presented with detailed analysis of the results. It has been concluded from the analysis that out of other deep-learning model [10] [43][44][45][46] for this thesis, YOLOv3 showed best classification performance, followed by Faster R-CNN and lastly by Tiny-YOLOv3.

5.1 Future Work

The autonomous text identification for arbitrary oriented languages is a topic where very few research works have been performed. There is a considerable difference between arbitrary oriented and regular language on the roads, as the environment is different, factors

that affect autonomous detection and identification will also be changed and can be particularly challenging[1], [47]. Therefore, future work can be done at the more scale and make improved versions of applications to make a direction of visually disabled as well as normal human being. In our thesis work, we have proposed an identification only for three languages i.e. English, Bengali, Hindi. But multilingual with more numbers of languages for detection as well as language translation with different RNN like LSTM could be great work in the near future [48].

References

[1] T. Q. Phan, P. Shivakumara, S. Tian, and C. L. Tan, "Recognizing text with perspective distortion in natural scenes," in *Proceedings of the IEEE International Conference on Computer Vision*, 2013, pp. 569–576. doi: 10.1109/ICCV.2013.76.

[2] K. Wang, B. Babenko, and S. Belongie, "End-to-End Scene Text Recognition."

[3] A. Brasoveanu, M. Moodie, and R. Agrawal, "Textual evidence for the perfunctoriness of independent medical reviews," in *CEUR Workshop Proceedings*, 2020, vol. 2657, pp. 1–9. doi: 10.1145/nnnnnnn.nnnnnnn.

[4] Y. Huang, Z. Sun, L. Jin, and C. Luo, "EPAN: Effective parts attention network for scene text recognition," *Neurocomputing*, vol. 376, pp. 202–213, Feb. 2020, doi: 10.1016/j.neucom.2019.10.010.

[5] C. Liu, C. Yang, and X.-C. Yin, "Open-set Text Recognition via Character-Context Decoupling," Apr. 2022, [Online]. Available: http://arxiv.org/abs/2204.05535

[6] B. Su and S. Lu, "Accurate Scene Text Recognition based on Recurrent Neural Network."

[7] A. Graves, A. Ch, S. Fernández, F. Gomez, J. Schmidhuber, and J. Ch, "Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks."

[8] Y. Gao, Y. Chen, J. Wang, and H. Lu, "Progressive rectification network for irregular text recognition," *Science China Information Sciences*, vol. 63, no. 2, Feb. 2020, doi: 10.1007/s11432-019-2710-7.

[9] C. Luo, L. Jin, and Z. Sun, "A Multi-Object Rectified Attention Network for Scene Text Recognition," Jan. 2019, [Online]. Available: http://arxiv.org/abs/1901.03003

[10] C. Yao, X. Bai, W. Liu, Y. Ma, and Z. Tu, "Detecting Texts of Arbitrary Orientations in Natural Images."

[11] T. Novikova, O. Barinova, P. Kohli, and V. Lempitsky, "Large-Lexicon Attribute-Consistent Text Recognition in Natural Images." [Online]. Available: http://algoval.essex.ac.uk/icdar/Datasets.html

[12] D. L. Smith, J. Feild, and E. Learned-Miller, “Enforcing Similarity Constraints with Integer Programming for Better Scene Text Recognition.” [Online]. Available: http://algoval.essex.ac.uk/icdar/Datasets.html

[13] A. Mishra, K. Alahari, and C. v Jawahar, “Top-Down and Bottom-up Cues for Scene Text Recognition.”

[14] S. Long, X. He, and C. Yao, “Scene Text Detection and Recognition: The Deep Learning Era,” Nov. 2018, [Online]. Available: http://arxiv.org/abs/1811.04256

[15] T. Wang *et al.*, “Decoupled Attention Network for Text Recognition,” Dec. 2019, [Online]. Available: http://arxiv.org/abs/1912.10205

[16] M. Liao *et al.*, “Scene Text Recognition from Two-Dimensional Perspective.” [Online]. Available: www.aaai.org

[17] Y. Gao, Y. Chen, J. Wang, and H. Lu, “Reading Scene Text with Attention Convolutional Sequence Modeling,” Sep. 2017, [Online]. Available: http://arxiv.org/abs/1709.04303

[18] S. Gunna, R. Saluja, and C. V. Jawahar, “Improving Scene Text Recognition for Indian Languages with Transfer Learning and Font Diversity,” *Journal of Imaging*, vol. 8, no. 4, Apr. 2022, doi: 10.3390/jimaging8040086.

[19] C.-Y. Lee and S. Osindero, “Recursive Recurrent Nets with Attention Modeling for OCR in the Wild.”

[20] “Paper_15_precision_comparision”.

[21] B. Shi, X. Bai, and C. Yao, “An End-to-End Trainable Neural Network for Image-based Sequence Recognition and Its Application to Scene Text Recognition,” Jul. 2015, [Online]. Available: http://arxiv.org/abs/1507.05717

[22] S. Maji, A. C. Berg, and J. Malik, “Efficient classification for additive kernel SVMs,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 1, pp. 66–77, 2013, doi: 10.1109/TPAMI.2012.62.

[23] L. Ballan, M. Bertini, A. del Bimbo, L. Seidenari, and G. Serra, “Event Detection and Recognition for Semantic Annotation of Video.”

[24] A. Mirza, O. Zeshan, M. Atif, and I. Siddiqi, “Detection and recognition of cursive text from video frames,” *Eurasip Journal on Image and Video Processing*, vol. 2020, no. 1, Dec. 2020, doi: 10.1186/s13640-020-00523-5.

[25] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks.” [Online]. Available: http://code.google.com/p/cuda-convnet/

[26] Z. Cheng, Y. Xu, F. Bai, Y. Niu, S. Pu, and S. Zhou, “AON: Towards Arbitrarily-Oriented Text Recognition.”

[27] J. Redmon and A. Farhadi, “YOLOv3: An Incremental Improvement,” Apr. 2018, [Online]. Available: http://arxiv.org/abs/1804.02767

[28] “What Is the Definition of Machine Learning? | Expert.ai | Expert.ai.” https://www.expert.ai/blog/machine-learning-definition/ (accessed Jun. 02, 2022).

[29] “Deep Learning.” https://www.deeplearningbook.org/ (accessed Jun. 02, 2022).

[30] O. Alsharif and J. Pineau, “End-to-End Text Recognition with Hybrid HMM Maxout Models,” Oct. 2013, [Online]. Available: http://arxiv.org/abs/1310.1811

[31] A. Coates *et al.*, “Text Detection and Character Recognition in Scene Images with Unsupervised Feature Learning.”

[32] C. Tian *et al.*, “IEEE TRANSACTIONS ON MULTIMEDIA 1 Coarse-to-Fine CNN for Image Super-resolution.” [Online]. Available: https://github.com/hellloxiaotian/CFSRCNN.

[33] H. Li, P. Wang, C. Shen, and G. Zhang, “Show, Attend and Read: A Simple and Strong Baseline for Irregular Text Recognition,” Nov. 2018, [Online]. Available: http://arxiv.org/abs/1811.00751

[34] H. Wang, Y. Yang, and B. Liu, “GMC: Graph-Based Multi-View Clustering,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 32, no. 6, pp. 1116–1129, Jun. 2020, doi: 10.1109/TKDE.2019.2903810.

[35] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition.” [Online]. Available: http://image-net.org/challenges/LSVRC/2015/

[36] Z. Cheng, Y. Xu, F. Bai, Y. Niu, S. Pu, and S. Zhou, “AON: Towards Arbitrarily-Oriented Text Recognition.”

[37] J. Liang, D. Doermann, and H. Li, “Camera-based analysis of text and documents: A survey,” *International Journal on Document Analysis and Recognition*, vol. 7, no. 2–3, pp. 84–104, Jul. 2005, doi: 10.1007/S10032-004-0138-Z.

[38] U. Roy, A. Mishra, K. Alahari, and C. v Jawahar, “Scene Text Recognition and Retrieval for Large Lexicons.”

[39] C. Yao *et al.*, “IEEE TRANSACTIONS ON IMAGE PROCESSING 1 Detecting Texts of Arbitrary Orientations in Natural Images 2 Index Terms,” 2012.

[40] L. Neumann, “Real-Time Scene Text Localization and Recognition.” [Online]. Available: http://cmp.felk.cvut.cz/

[41] C. Yao, X. Bai, B. Shi, and W. Liu, “Strokelets: A Learned Multi-Scale Representation for Scene Text Recognition.”

[42] X. Yue, “RobustScanner: Dynamically Enhancing Positional Clues for Robust Text Recognition.”

[43] T. E. de Campos, B. Rakesh Babu, and M. Varma, “CHARACTER RECOGNITION IN NATURAL IMAGES.” [Online]. Available: http://www.ee.surrey.ac.uk/CVSSP/demos/chars74k/

[44] T. Wang, D. J. Wu, A. Coates, and A. Y. Ng, “End-to-End Text Recognition with Convolutional Neural Networks.”

[45] S. M. Lucas, A. Panaretos, L. Sosa, A. Tang, S. Wong, and R. Young, “ICDAR 2003 Robust Reading Competitions,” 2003.

[46] A. Graves, M. Liwicki, S. Fernández, R. Bertolami, H. Bunke, and J. Schmidhuber, “A Novel Connectionist System for Unconstrained Handwriting Recognition.” [Online]. Available: www.e-Beam.com

[47] T. Novikova, O. Barinova, P. Kohli, and V. Lempitsky, “Large-Lexicon Attribute-Consistent Text Recognition in Natural Images.” [Online]. Available: http://algoval.essex.ac.uk/icdar/Datasets.html

[48] H. Ren, W. Wang, and C. Liu, “Recognizing online handwritten Chinese characters using RNNs with new computing architectures,” *Pattern Recognition*, vol. 93, pp. 179–192, Sep. 2019, doi: 10.1016/J.PATCOG.2019.04.015.