Master of Science in Computer Science
June 2023

Real time Optical Character Recognition in steel bars using YOLOV5

Monica Gattupalli

Faculty of Computing, Blekinge Institute of Technology, 371 79 Karlskrona, Sweden

This thesis is submitted to the Faculty of Computing at Blekinge Institute of Technology in partial fulfillment of the requirements for the degree of Master of Science in Computer Science. The thesis is equivalent to 20 weeks of full-time studies.

The authors declare that they are the sole authors of this thesis and that they have not used any sources other than those listed in the bibliography and identified as references. They further declare that they have not submitted this thesis at any other institution to obtain a degree.

Contact Information:
Author:
Monica Gattupalli
E-mail: moga20@student.bth.se

University advisor:
Dr. Hüseyin Kusetoğulları
Department of Computer Science

Industrial Supervisor 1:
Krister Ekström
Swerim AB ,Stockholm, Sweden

Industrial Supervisor 2:
Privschek Edwin
Ovako Sweden AB, Hofors, Sweden

Faculty of Computing
Blekinge Institute of Technology
SE–371 79 Karlskrona, Sweden

Internet : www.bth.se
Phone : +46 455 38 50 00
Fax : +46 455 38 50 57

Abstract

Background. Identifying the quality of the products in the manufacturing industry is a challenging task. Manufacturers use needles to print unique numbers on the products to differentiate between good and bad quality products. However, identifying these needle printed characters can be difficult. Hence, new technologies like deep learning and optical character recognition (OCR) are used to identify these characters.

Objective. The primary objective of this thesis is to identify the needle-printed characters on steel bars. This objective is divided into two sub-objectives. The first sub-objective is to identify the region of interest on the steel bars and extract it from the images. The second sub-objective is to identify the characters on the steel bars from the extracted images. The YOLOV5 and YOLOV5-obb object detection algorithms are used to achieve these objectives.

Method. Literature review was performed at first to select the algorithms, then the research was to collect the dataset, which was provided by OVAKO. The dataset included 1000 old images and 3000 new images of steel bars. To answer the RQ2, at first existing OCR techniques were used on the old images which had low accuracy levels. So, the YOLOV5 algorithm was used on old images to detect the region of interest. Different rotation techniques are applied to the cropped images(cropped after the bounding box is detected) no promising result is observed so YOLOV5 at the character level is used in identifying the characters, the results are unsatisfactory. To achieve this, YOLOV5-obb was used on the new images, which resulted in good accuracy levels.

Results. Accuracy and mAP are used to assess the performance of OCRs and selected object detection algorithms. The current study proved Existing OCR was also used in the extraction, however, it had an accuracy of 0%, which implies it failed to identify characters. With a mAP of 0.95, YOLOV5 is good at extracting cropped images but fails to identify the characters. When YOLOV5-obb is used for attaining orientation, it achieves a mAP of 0.93. Due to time constraint, the last part of the thesis was not implemented.

Conclusion. The present research employed YOLOV5 and YOLOV5-obb object detection algorithms to identify needle-printed characters on steel bars. By first selecting the region of interest and then extracting images, the study objectives were met. Finally, character-level identification was performed on the old images using the YOLOV5 technique and on the new images using the YOLOV5-obb algorithm, with promising results.

Keywords: *Deep learning, Object detection, Tesseract OCR, YOLOV5, YOLOV5-obb.*

Acknowledgments

I would like to express my profound gratitude to Blekinge Tekniska Högskola, Ovako Sweden AB, and Swerim AB for providing me with the invaluable opportunity to work with them.

Furthermore, I want to express my heartfelt gratitude to the people who played critical parts in my thesis journey. Dr. Hüseyin Kusetoullar, my thesis supervisor, and my industrial supervisors, Krister Ekström and Privschek Edwin, deserve special gratitude for their consistent assistance and feedback during my thesis work. Their continual encouragement and helpful input were invaluable in guiding me through any difficulties that came during my research.

Furthermore, I would like to recognize and thank my parents for their constant moral support throughout my time at Blekinge Tekniska Högskola.

Contents

List of Figures

List of Tables

List of Abbreviations

CNN	Convolutional Neural Network
CRNN	Convolutional Recurrent Neural Network
FPS	Frames per second
IOU	Intersection over union
LSTM	Long Short-Term Memory
mAP	Mean Average Precision
NMS	Non-Maximum Suppression
OCR	Optical Character Recognization
OEM	OCR Engine Mode
PSM	Page Segmentation Mode
R-CNN	Recurrent Convolutional Neural Network
ReLU	Rectified Linear Unit
RNN	Recurrent Neural Network
ROI	Region of Interest
SSD	Single Stage Detector
YOLO	You Only Look Once
YOLOV5	You Only Look Once Version5
YOLOV5-obb	You Only Look Once Version5 -oriented bounding box

Chapter 1

Introduction

It is impossible to imagine a future without industries, which have grown exponentially during the previous few decades. Most sectors are focusing on product quality and development through the use of new technology. Many traditional methods are being phased out in favor of cutting-edge procedures, which improves industry quality and revenues. Product numbers, digital QR codes, barcodes, and written characters on the objects that aid in distinguishing between high-quality and low-quality products can all be utilized to differentiate the products. These printed characters can be photographed or remembered.

The identification and recognition of text on photographs or camera-captured photos is a well-researched problem in computer vision, sometimes known as optical character recognition (OCR). In general, image data is unstructured, making it challenging to translate and recognize images as digital text. Using a standard machine learning technique also makes achieving high item detection accuracy difficult [51], [8]. Many Deep Learning text identification approaches have demonstrated promising results with greater accuracy and efficiency when compared to machine learning algorithms.

OCR is a complex process that extracts text from images and converts it to digital form using a range of learning and modeling techniques. Traditional OCR recognizes text on fixed areas in photos (such as license plates), but the images might differ in form, brightness, orientation, and other ways. To reply to various types of photographs, the robust algorithm must be modified. However, object detection and text detection have distinct concerns and difficulties [51]. Various approaches, like R-CNN and SSD (single-stage detectors), are used to cope with realistic images [28].

For object detection, the most recent method, YOLO (You Only Look Once), is highly recommended. Prior to 2015, a variety of deep learning techniques, such as R-CNN, fast R-CNN, and SSD [28], were employed to recognize objects, but they were slow and ineffective. To date, eight versions of YOLO have been launched, with each one undergoing substantial enhancements and changes. YOLO version 4 was written in C, while version 5 was written in Python [3].

YOLO takes a photo as input, runs it through all of the algorithms' convolutional layers, then outputs the result with a bounding box. YOLO uses the Intersection over union (IOU) technique to generate a single bounding box rather than several bounding boxes based on the number of convolution layers [19].

Text detection, along with object detection, is a traditional difficulty in this period. In some applications, both object detection and text detection must be done simultaneously. Many OCR systems for text detection have been developed and are available in free versions, including Tesseract, ABBYY FineReader, Google Docs OCR, and Transym. All of these OCR services may be used on scanned documents, camera-acquired images, written documents, and so on [48].

The main objective of this thesis is to investigate the best OCR method for resolving the practical issue. In OVAKO, it is necessary to identify the characters on the steel bars. Using the advancements in deep learning in object and text recognition, this thesis investigates the best OCR techniques that accurately predict the characters on the steel bars. It also seeks to pinpoint the uncertainty conditions that still require improvement to achieve accurate text prediction on the steel bars. The company provides real-world data for the investigation as its data source.

1.1 Problem Definition

Maintaining traceability of unique manufactured units is critical for manufacturing quality control. This entails giving each object a unique identification mark, which is usually a stamped number on the material. Reading these marks, however, can be difficult, especially with fluctuating image quality.

Deep Learning-based image analysis improvements have provided a solution. With enough training data, these algorithms can be exceedingly resilient. Some methods include "transfer learning," which involves first training on a broad dataset to gain general knowledge and then fine-tuning for specific images. This allows for the automatic and reliable reading of identification markers, ensuring effective industrial traceability.

.

1.1.1 Image data

All the data provided by the company is in image format and needle printed on them. Typically, background noise will be present in the photos along the region of interest. The steel bars images from the company feature several problems and many places that need improvement when compared to normal images.

Figure 1.1 is the combination of different types of images like scanned document, which has text on them captured with a camera, Figure 1.2is the steel bar image which is needle printed and captured with a camera in the production unit.

Upon comparing Figure 1.1 with Figure 1.2, we observe that Figure 1.2 has much background noise like rough edges on the steel bar, and extra background other than the steel bar, due to the dark green on the steel rods the character is a little hard to identify with the necked eye, we can also see there are patterns on the steel bars which make the text identification much harder. The light focus is constant in all images. Sometimes the shadow of the rough edges will affect the text on the steel

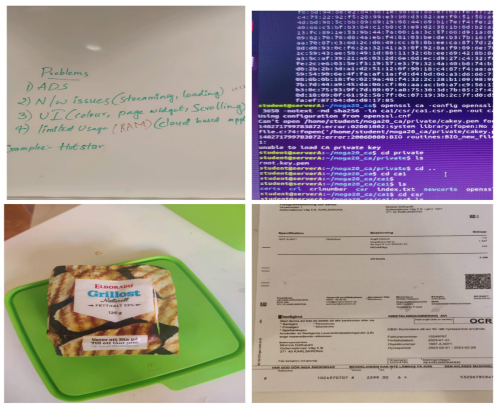

Figure 1.1: Normal Images

Figure 1.2: steel bars

bar. The characters are printed with a needle so, there will be variations of sharpness to the needle which will affect the marking on the steel bars. The steel bars will be on the rolling mill, thus there will be a rotation where the rods will occasionally go close to the needle and occasionally away from the needle which causes the causing a difference in the depth of the text on the steel bars. Due to the rotation of the steel bar at 360 degrees, not every image of a steel bar will be in the same orientation.

By considering all the aspects of Figure 1.2, it is important that the algorithms selected should be good enough to localize the text position without losing any kind of data or information from the image.

1.2 Aims and Objectives

Aim: This project aims to investigate and implement a robust text classification algorithm that is preferably as accurate as a human reader in reading the text on the ends of the steel bars.

Objectives:

1. Detection and localization of text on steel bars, resulting in images with bounded boxes.

2. Extracting text from the images produced by objective 1.

3. Report success rates and uncertainty estimates broken down by marking/surface/image quality.

1.3 Research Questions

RQ1: What are the best-performing object detection algorithms in detecting characters on steel bar ?

Motivation: The RQ1 is quite important to selecting the best algorithm for answering the industrial problem. This RQ1 needs a mini literature review to identify the algorithms that actually suites for the problem.

RQ2:How good is the proposed model at recognizing the characters on the steel bars?

Motivation: The RQ2 is quite important to determine how well our model is detecting the text on the steel bars, the accuracy of the model helps to justify whether the proposed model is a good fit or not.

RQ3: What are the uncertainty conditions in the images that must be overcome to make better predictions?

Motivation: This RQ3 helps the company to understand the exact conditions, that need to be improved in the aspects of marking, and camera quality to get the better recognition of the text.

1.4 Ethical Aspects

Ethical considerations are central to this experiment, and they are carried through the thesis. OVAKO's picture dataset and information regarding steel production are treated with strict confidentiality. The research's ethical commitment and collaborative integrity are underscored by data security and respect for proprietary information.

1.5 Outline

The thesis report comprises several chapters that address the problem of object detection in computer vision and the research conducted to solve the industrial problem of "OVAKO".

- Chapter 1 introduces the topic, highlighting the role of computer vision algorithms and advancements in this field. The chapter also defines the problem statement and outlines the research objectives and research questions.
- Chapter 2 provides background information about existing OCR techniques, annotating tools, and major object detection techniques.
- Chapter 3 focuses on related work in the field of object detection algorithms, deep learning, and optical character recognition. This chapter discusses various research works conducted in these areas.
- Chapter 4 outlines the literature review, data collection method, and methodology employed for old and new images. This chapter details the approach taken to collect data for the study.

- Chapter 5provides a deeper analysis of the methodology and results chapters and provides an in-depth view of the research findings
- Chapter 6 focuses on the results and the Analysis of the Industrial problem which focuses on the
- Chapter 7 is the conclusion and future work which concludes the work and gives the different ways can this research be extended.

Chapter 2

Background

This section includes background information on existing methods for optical character recognition, machine learning, deep learning, CNN, object identification, mAP, image annotations, YOLOV5, and YOLOV5-obb.

2.1 Existing solutions for OCR

OCR technology is widely used to recognize text within an image or document and convert it into machine-readable text. Many OCR techniques have been developed to solve different OCR problems and new text detectors are being developed to address new industrial challenges. Here are some of the existing OCR techniques:

2.1.1 Text Recognition Engines

OCR engines have been designed to scan and extract text from a variety of sources, including papers, pictures, and other mediums. Among these OCR engines, Tesseract OCR, Easy OCR, Google Vision OCR, and Google Lens stand out as useful tools for text recognition and extraction.

Tesseract OCR and Easy OCR have been chosen as the OCR engines of choice for text detection in the context of the project at hand. These two OCR engines have acquired popularity due to their availability as open-source solutions and their ability to recognize and extract text from a variety of sources.

Tesseract OCR [47] is a popular open-source OCR engine created by Google. It has grown over time and is well-known for its powerful text recognition capabilities. Tesseract OCR excels in handling text in a variety of fonts, sizes, and styles. Tesseract OCR holds 2 different parameters "Page segmentation Mode(psm)" and "OCR Engine Mode(oem)" which are responsible for considering the format of the text and the internal design architecture of Tesseract OCR.

Easy OCR [18] is another open-source OCR engine that has gained popularity due to its user-friendliness and ease of integration. It is intended to make the process of implementing OCR for text detection tasks easier, making it a viable option for developers.

In this thesis, Tesseract OCR and Easy OCR were used to perform text detection and the results are discussed in the Results chapter.

2.1.2 Mobilenet-SSDv2

The object detection model developed in 2018 using the TensorFlow framework, has a complex architecture with 267 layers. This is a one-shot detector with two critical components: the SSD layer and Mobilenet-v2. The SSD layer is excellent at recognising objects of varying sizes within a single network pass, making it perfect for real-time applications. Meanwhile, the design is supplemented by Mobilenet-v2, which is noted for its computing efficiency and accuracy balancing. This model is notable for its ability to do both object localization and picture classification at the same time, making it adaptable and well-suited for Optical Character Recognition (OCR) tasks. Indeed, it has been used in OCR applications, as indicated by citations in previous studies [54] [10] [41] [28].

2.1.3 CRNN

CRNN is a deep learning technique designed for text recognition. It is a combination of convolution neural networks (CNN) and recurrent neural networks (RNN) that extract features and pass them to long short-term memory (LSTM), a language model that helps mAP the labels and identify characters by a dense layer [33].

Due to the RNN architecture, a single character can be detected repeatedly, to address this, a loss function is needed at the end of the CRNN to focus on the output. A solution was developed by Graves et al. in 2006, which is known as CTS.

2.2 Machine Learning

Machine learning is a subset of artificial intelligence that entails the development of algorithms and models that enable computer systems to improve their performance on specific tasks by using data inputs. This technique, rather than being explicitly coded, allows systems to learn how to make predictions or decisions by being exposed to situations [14].

There are three types of machine learning: supervised, unsupervised, and reinforcement learning [14]. Unsupervised learning looks for patterns in unlabeled data, whereas supervised learning trains a model on labeled data. Reinforcement learning entails learning to make decisions based on feedback from the environment.

Machine learning has several applications, including natural language processing, picture recognition, fraud detection, and recommendation systems. It has the ability to significantly improve processes by increasing efficiency and accuracy.

2.2.1 Supervised Learning

Supervised learning is a sort of machine learning in which the algorithm is trained on a labeled dataset. In this sort of learning, the algorithm learns to create predictions or judgments based on input and output data. The algorithm gets input data and uses the output data to train itself to make the right predictions [46].

1. **Classification** A categorical output variable, such as categorizing images or forecasting whether a buyer will buy a product.

2. **Regression** A continuous output variable, such as forecasting the price of a house based on its qualities or product demand.

2.2.2 Unsupervised Learning

Unsupervised learning is a type of machine learning in which the algorithm is taught on an unlabeled dataset. The algorithm learns to recognize patterns and relationships in the input data in the absence of any specified output variable to forecast. The program attempts to group similar instances or uncover qualities that describe the underlying structure of the data [40]. .

1. **Clustering** The algorithm groups similar instances together based on their attributes.

2. **Dimensionality Reduction** occurs when the algorithm reduces the number of characteristics in the data while maintaining as much information as possible.

2.3 Reinforcement Learning

Reinforcement learning is a type of machine learning in which the algorithm learns to make decisions based on feedback from the environment. In this type of learning, the algorithm interacts with its surroundings and receives feedback in the form of rewards or penalties based on its activities. The goal of the algorithm is to learn how to execute actions that maximize overall reward over time [43].

2.4 Deep Learning

Deep learning is a type of machine learning in which neural networks are used to learn features and patterns without explicit instruction. There are at least three layers in neural network architecture: the input layer, the hidden layer, and the output layer [42].

The input layer receives input data, which in image recognition tasks is often the image size. Hidden layers are in charge of learning and extracting features from input data, and they can be customized with different activation functions, units, batch normalization, and max pooling layers. Depending on the kind of classification, such as binary or multi-class classification, the output layer employs a different activation function [42].

During the training phase, a huge amount of data is fed into the network, allowing the network to automatically alter the weights. This allows the network to learn every feature of the input data, boosting its classification and prediction abilities.

Deep learning has numerous applications in a variety of industries, including healthcare, manufacturing, and product-based industries. It has evolved into a formidable instrument for tackling complicated problems that necessitate enormous amounts of data and great degrees of precision.

2.4.1 Convolutional Neural Network (CNN)

Convolutional neural networks are a type of network that excels at handling images and videos. The network receives photos or video as input, which are then processed by various internal layers that extract and store the information from the input [52].

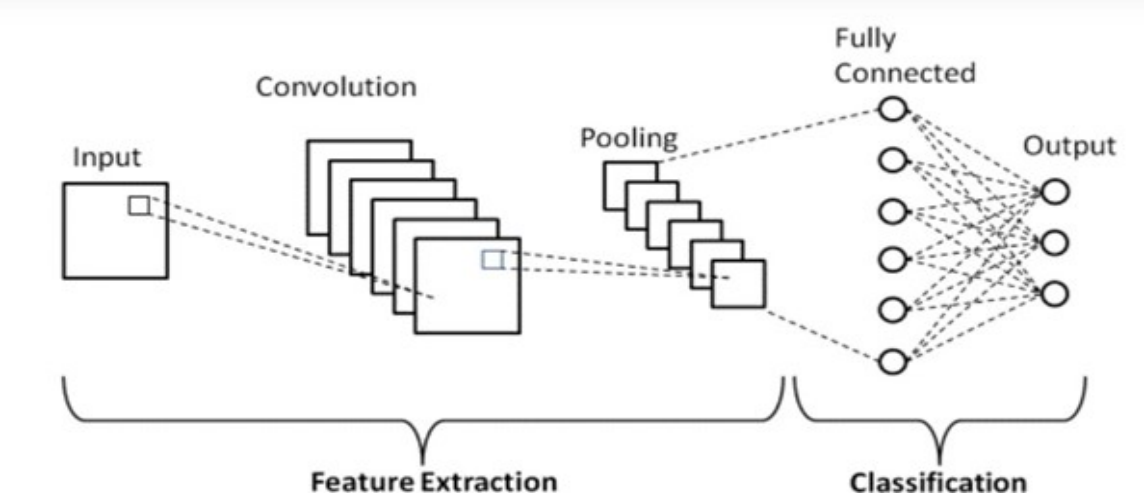

Figure 2.1: Overview of CNN architecture [45].

Figure 2.1 is the design of basic CNN architecture [52], with the first layer being convolutional layers, which take an image as input and apply a set of filters to it to extract features. During the training phase, CNN automatically learns features and adjusts weights in the neural network.

The convolution is followed by the pooling layer. Pooling layers might be max, min, or sum depending on the needs of the user. The pooling layer aids in decreasing the output of the convolution layer and the feature mAP, hence avoiding network complexity and overfitting.

2.5 Object Detection

A computer vision technique that recognizes and locates things of interest inside an image or video frame is known as object detection. This approach is used in a wide range of applications, including self-driving cars, surveillance systems, and augmented reality [4].

Techniques for detecting objects are frequently divided into two stages: region suggestion and classification. The location suggestion stage finds potential object placements within an image by using approaches such as selective search or area proposal networks. Each suggested region is characterized as having or not having an object of interest during the categorization step. Deep learning-based techniques, such as convolutional neural networks (CNNs), are typically used to do this.

Three popular object identification algorithms include faster R-CNN, YOLO (You Only Look Once), and SSD (Single Shot Detector). These algorithms differ in terms of speed and accuracy, as well as application applicability [20].

2.5.1 Mean Average Precision(mAP)

The Mean Average Precision (mAP) metric is widely used to evaluate item identification and other information retrieval tasks. It assesses the accuracy and completeness of a system's predictions.

mAP is calculated using the Average Precision (AP) for each class, which is the area under the precision-recall curve for that class. The precision-recall curve is produced by altering the detection threshold and computing the accuracy and recall values for each threshold. The ratio of true positives to total predicted positives is described as precision, and the ratio of true positives to total real positives is defined as recall [44]

The mAP is calculated as the mean of the AP values across all classes after determining the AP for each class. In other words, mAP takes into account the system's prediction accuracy for each class, as well as the balance of precision and recall.

The Formula of mAP [32]:

$$mAP = \frac{1}{C} \sum_{c=1}^{C} AP_c \tag{2.1}$$

Where:

$$\begin{aligned} mAP &: \text{Mean Average Precision} \\ C &: \text{Total number of classes} \\ AP_c &: \text{Average Precision for class } c \end{aligned}$$

The mAP metric can be used to compare several object detection models or methods, as well as to track the performance of a single model over time. It produces a single, easily interpretable score that reflects the system's overall accuracy and prediction completeness [44].

2.5.2 Annotating Tools

Annotating tools are a kind of application that is used to annotate different forms of data for different machine learning and deep learning problems. These technologies are widely utilized in domains such as computer vision, natural language processing, and others where labeled data is essential for training and evaluating models.

2.5.2.1 Makesense.ai

Makesense.ai is an easy-to-use web platform that allows you to classify images for free. You don't need to do anything complicated; simply go to the website and start utilizing it. It works on any operating system, so you may use it on any computer. This tool is especially useful for modest projects requiring computer vision and deep learning. It makes preparing your dataset more quicker and easier. When you're finished labeling, you may effortlessly download your labels in a variety of supported formats. The tool was written in TypeScript and is based on the React/Redux framework, making it speedy and dependable [30].

2.5.2.2 Roboflow

Roboflow is a platform that provides data management, annotation, and preparation tools and services for machine learning and computer vision projects. It is intended to ease data preparation, making it easier to train and deploy machine learning models, notably in computer vision [38].

Data annotations, data augmentation, annotation conversion, model training, and model deployment are the core features of Roboflow. Roboflow is accessible in a free version that can be viewed straight online. This application allows us to keep track of work and allows users to create versions of their work. The paid version of the roboflow provided the advanced option that allows storing the trained models and no code model training can be seen.

2.5.3 Image Annotations

Annotations are frequently used by object detection algorithms to locate and identify items inside an image. Annotations can in a variety of forms, including bounding boxes, polygonal segmentation, 3D cuboids, and lines [37].

1. **Rectangular bounding box:** Bounding boxes are rectangular annotations that indicate the image's x- and y-axis coordinates. They are the most common kind of annotation. The bounding boxes in Figure 2.2 are rectangular in shape

Figure 2.2: Rectangular Bounding Box.

   and enclose the ROI. It is crucial to note, however, that these rectangular bounding boxes do not always match the exact shape of the items included therein; rather, they give a rectangle region that contains the object of interest.

2. **Polygonal Annotations:** Polygonal segmentation annotations are used for items such as humans and cars that cannot be correctly characterized by a rectangle. These annotations are exactly suited to the shape of the object. The annotations in Figure 2.3 take the form of polygonal forms for the bounding boxes. The polygonal annotations in this picture accurately portray the curves and shape of steel bars while conforming to the uneven boundaries of the object.

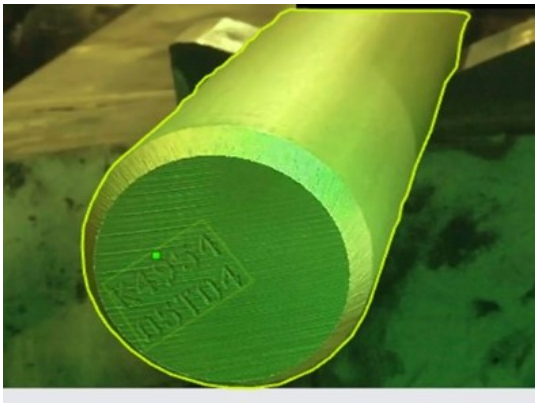

Figure 2.3: Polygon Annotations.

Figure 2.4: Cuboid Annotations [7].

3. **3D cuboids:** 3D cuboid are similar to rectangular bounding boxes in that they determine the depth of the item.

   Figure 2.4 shows that the annotations are in the shape of cuboids. These cuboid annotations are used to capture the car's depth as well as its length and breadth. They essentially provide a three-dimensional picture of the car's shape, including its height, which provides more insights of the object.

4. **Line Annotations:** These are used for photographs containing linear features, such as roads.

   Figure 2.5shows line annotations that go along the object's height. These lines run parallel to the item and effectively outline its vertical extent.

The below Subsections 2.6,2.7 and 2.8 are the algorithms used in the current study which are selected from the literature review which is stated in Section 4.1.

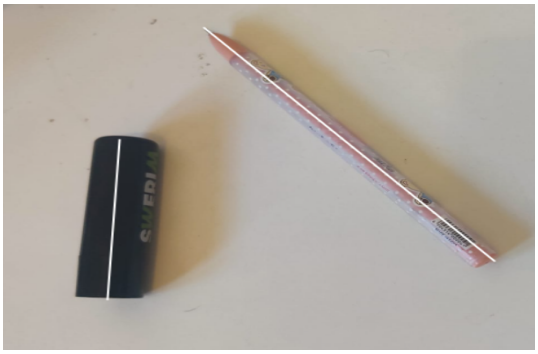

Figure 2.5: Line Annotations.

2.6 You Only Look Once(YOLO)

YOLO (You Only Look Once) is a prominent object detection method that is noted for its speed and accuracy. Joseph Redmon debuted it in 2015, and it has since gone through eight different versions, with major improvements and updates to its functionality and architecture [53]. Figure 2.6 is the architecture of YOLO with 24

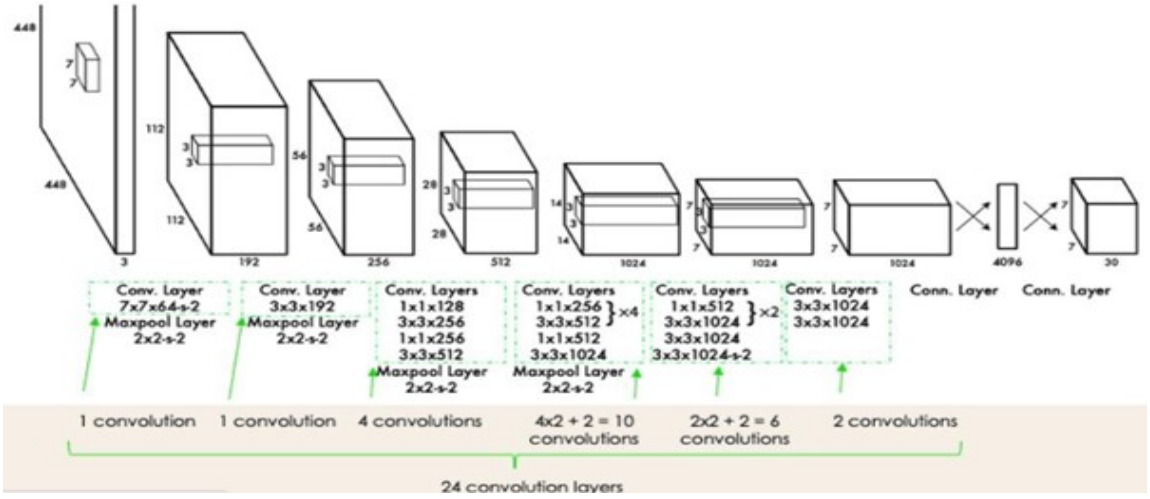

Figure 2.6: YOLO Architecture [9].

convolutional layers, four max-pooling layers, and two fully connected layers used by YOLO. YOLO is intended to process photos with a resolution of 448 x 448. The image is scaled and then processed by the first convolutional layer, which minimizes the number of channels. The first convolutional layer's output is then processed through a 3X3 convolutional layer, producing a cuboidal output.

YOLO employs the Rectified Linear Unit (ReLU) activation function in all layers to add nonlinearity into the network. The final layer of the YOLO network employs a linear activation function to produce the final output. The linear activation function is a basic function that returns the input exactly as it is. In the case of object detection, the result is a set of bounding boxes representing the items recognized in the image [2].

A convolutional neural network powers YOLO's underlying architecture, allowing

it to detect objects in pictures, videos, or real-time. As input, YOLO accepts an image and a text file giving the bounding box coordinates of the objects, as well as their class names. This information is then passed to the convolution neural network, which uses it to detect the objects in the input.

YOLO divides the input image into grids and finds the bounding boxes as well as the confidence score using the Intersection over Union (IOU) technique. YOLO guesses the bounding boxes using this method. Multiple bounding boxes for a single object are commonly identified in object detection. To avoid this, YOLO employs a post-processing technique known as Non-Maximum Suppression (NMS), which helps to enhance accuracy and reduce redundant bounding boxes [20].

YOLO offers an extensive list of internal settings that allow users to acquire the desired output format. It also computes frames per second (FPS) to assess the rate at which objects are detected. The original version of YOLO was trained on 224X224 images and the PASCAL VOC dataset, which contains 20 objects. However, each edition of YOLO has experienced substantial adjustments, and it is now capable of handling many problems with more precision and speed.

2.7 YOLOV5

YOLOV5 is a quick and user-friendly object detection system built with the pyTorch framework and trained on the COCO dataset of 80 classes. The YOLOV5 architecture is divided into five models: YOLOV5n, YOLOV5s, YOLOV5m, YOLOV5l, and YOLOV5xl.Each had a distinct mAP and FPS and was trained on a 640*640 image. YOLOV5 also includes a p6 series that was trained on larger images (1280*1280), known as YOLOV5n6, YOLOV5s6, YOLOV5m6, YOLOV5l6, and YOLOV5x6 [53].

YOLOV5, like other object detection systems, recognises things in images using annotations. In YOLOV5, the annotation format is **"object-class> x> y> width> height>"**, where the object-class denotes the name of the object class and x and y provide the bounding box coordinates. The width and height of the rectangular bounding box are indicated by the final two parameters. YOLOV5 can accurately locate and identify things in a picture by using this annotation format.

In order to use YOLOV5, input images and text files are required, as well as a YAML file containing dataset metadata such as the paths to train, validate, and test images and the number of classes in the dataset. YOLOV5 allows users to alter hyperparameters based on data utilisation, allowing for greater training flexibility. Furthermore, YOLOV5 supports the storage of test data bounding boxes as well as image cropping depending on the detected bounding boxes.

YOLOV5 initially uses the weights of YOLOV5s to train the data throughout the training process, with the option for the user to select weights from any other version based on their data and mAP. YOLOV5 can be cloned from the official git repository [50] and used for a variety of purposes.

2.8 YOLOV5-OBB

On March 12, 2020, YOLOV5-obb was released as an enhanced version of the YOLOV5 object detection method. It adds support for oriented bounding boxes (OBBs), which better match objects positioned at angles. YOLOV5-obb's underlying architecture is based on a CNN and was created with the PyTorch deep learning framework [1].

YOLOV5-obb uses the annotations format (x1, y1, x2, y2, x3, y3, x4, y4, xi, yi, name, difficulty) [20]. The first eight parameters (x1, y1, x2, y2, x3, y3, x4, y4) are the oriented bounding box coordinates. The next two parameters (xi, yi) are used to store any additional information about the bounding box. The name argument specifies the name of the class of the object being recognized, while the difficulty parameter, with a value of 0 or 1, indicates the difficulty degree in recognizing the object.

CUDA is necessary in the background to execute YOLOV5-obb. This is due to the fact that without CUDA, NMS parameters cannot be loaded. NMS is a technique used in object detection algorithms to suppress numerous detections of the same object in a picture.

YOLOV5-obb is a substantial breakthrough in object detection, especially for situations where objects may be positioned at an angle. Its use of oriented bounding boxes enables more accurate and precise object identification, and its integration with CUDA allows for rapid and efficient processing of massive amounts of data.

Chapter 3

Related Work

This chapter briefly discusses the previous works and overview of the different research articles on object detection algorithm YOLO and text detection, OCR techniques, and tools used in annotating the images for object detection.

Haifeng Wang, Changzai Pan, Xiao Guo, Chunlin Ji, and Ke Deng et al., [51] discussed various deep learning algorithms associated with object detection, as well as key points related to OCR. The statistical model in the field of text detection and recognition, as well as how the deep learning era has changed object detection since 2010. Many algorithms, such as CNN, R-CNN, Fast RCNN, Faster-RCNN, YOLO, and SSD, have been developed to detect objects in images/videos. Many software packages, such as Detectron2, Tesseract, and CRNN, are used in text detection, and they eventually concluded with different applications that require both object and text detection.

Peiyuan Jiang, Daji Ergu, Fangyao Liu, Ying Cai, and Bo Ma et al ., [19] reviewed the YOLO object detection technique and versions. They also compared CNN with the YOLO algorithm. Major advancements between YOLO versions are also discussed. When comparing YOLO versions, they concluded that YOLO V5 is more flexible, as it employs the Hardswish activation function.

Myung-Cheol, Ju-young et al., [39] Lee proposed RF-RCNN, a faster RCNN architecture with a refining block. The refining block is made up of the regressor and the classifier. Humans and license plates are detected using the proposed architecture. The new architecture outperformed faster RCNN in detecting humans and license plates.

Xingyu Zeng, Wanli Ouyang, Bin Yang, Junjie Yan, and Xiagang Wang et al [55] proposed gated bi- directional CNN (GBD-Net) for object detection. This GBD-Net can be merged after the convolution layer in any architecture to help retain the features of the region of interest and images in different resolutions. The proposed methodology has been validated using ImageNet, Pascal VOC2007, and Microsoft COCO.

Rayson Laroca, Evair Severo, Luiz Zanlorensi , Luis S. Oliveria, Gabriel Resende Goncalves, William Robson Schwartz ,David Menotti et al., [23] has used the YOLO object detector to identify license plates, and trained the network using 4500 fully

annotated photos from 150 automobiles using two different CNNs. Up to 6 characters on the license plate may be recognized by the suggested model with an accuracy of 93.53%.

Jian Han , Yaping Liao, Junyou Zhang, Shufeng Wang and Sixian Li et al., [16] has employed YOLO to identify color and LiDAR depth pictures. The suggested model is broken down into two parts. In the first step, the YOLO method is applied to the images, in second step convolution and pooling layers are applied. To obtain the precise object detection, the bounding boxes from two phases are combined.

Jia-Ping Lin, Min-Te Sun et al ., [26] has designed a technique for counting vehicles. The vehicle counter is in charge of counting after the YOLO algorithm identifies the vehicle and creates bounding boxes around the objects, these bounding box co-ordinates are then saved in the buffer. The video is provided as the YOLO algorithm's input. When trained with the COCO dataset, YOLO typically recognizes 80 moving things. Therefore, they modified YOLO to only detect the car, bus, and truck while masking out all other moving objects. The studies use footage taken in the morning, the afternoon, and the evening, and all three videos are made with 100% accuracy.

Rachel Huang, Jonathan Pedoeem, Cuixian Chen et al ., [17] has created the real-time YOLO-LITE object detection system, which was trained using the PASCAL VOC and COCO datasets. Personal computers can be used to run the model. YOLO-LITE performed well on a small system, averaging 21 frames per second. They also talked about how to make the existing system better in the future.

Wei Liu, Dragomir Anguelov, Dumitru Exhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, Alexander C.Berg et al ., [28]has proposed a new model for detecting objects which is named as Single Shot Multibox Detector SSD and this model is experimented upon PASCAL VOC, ILSVRC and COCO datasets which showed the promising results and justified that SSD is good at object detection.it also achieved 74.3% mAP on 300X300 input size, when 512 X 512 input is given it outperformed than Faster RCNN with 76.9% mAP .

Michal Buta, Luka Neumann, Jiri Matas et al ., [8] has proposed a scene text detector which is able to detect the text in a video stream. It can also detect the different languages like Latin, Hebrew and Chinese along with English. Designed detector is good at detecting the text with have stroke ending, but it failed to detect text which have no corner. They concluded that proposed model detects the 25% more characters than the MSER and 3 times faster than the MSER detector .

Yunong Tian, Guodong Yang, ZheWang, Hao Wang, En Li, Zize Liang et al ., [49] has developed an real time detection system which detects the growth of the apple developed using YOLO-V3. Images of immature apples, expanding apples, and ripples are initially gathered, then data is increased using augmentation techniques. The YOLO -V3 feature layers are employed with the DenseNet technique, which improved network performance. Comparing the performance of the YOLOV3-dense

net model to that of the YOLOV3 and Faster R-CNN with VGG-16 model.

Chirag Patel, Atul Patel, Dharmendra Patel et al ., [36] conducted a comparison between the Transym OCR and Tesseract OCR. Text detection is performed on images in both color and grayscale. Tesseract produced text detection accuracy of 61% and 70% on color and grayscale photos, respectively. It also performed better than Transym, taking 1 second and 0.82 seconds to process color and grayscale images, respectively.

Malathi T, Selvamuthukumaran D, Diwaan Chandar C S, Niranjan V, Swashthika A k et al., [31]has performed an comparative study between Microsoft OCR and tesseract OCR on different type of images which have white background, dark background, different fonts, colored font, grey scaled in most of the cases Microsoft OCR provided better results than tesseract OCR, but in some cases vice versa, but when coming to time of recognition tesseract OCR is better.

Rifiana Arief, Achmad benny Mutiara, Tubagus Maulana Kusuma, Hustinawaty et al ., [6]applied Google vision OCR on the document image to recognize the text in Hadoop environment. The OCR library is 2 times faster when running in Hadoop environment also extract the exact text by reaching the 100% accuracy.

Xin Li, Zhenhua Cai, Xi Zhao et al ., [25]has proposed a upgraded architecture of YOLOV5 which is named as oriented-YOLOV5 which is able to do the angular predictions. Circular smooth Label (CSL) technique is integrated into YOLOV5 which is responsible for the angle detections. CSL is integrated with the other models of YOLOV5 like YOLOV5s, YOLOV5m, YOLOV5l which achieved the mAP of 68.86% , 70.03%,71.2% respectively. Finally, they concluded that oriented-YOLOV5 is good at detecting the vertical view or object that is placed at an angle.

Qing Wen, Rong Liu, Bio Ao, Kuan Li et al., [12] has performed the experimental study on Aerial image object detection based on improved YOLOV5 (integrated the CSL and YOLOV5) which helped in detecting the rotated objected. performed many pre-processing steps to make algorithm work better. They also compared the results of improved YOLOV5 with the models of YOLOV5 and observed new version of YOLOV5 performs better than the original version of YOLOV5, as a future enhancement they also stated that focusing on encoding for CSL will achieve the better results [42].

CHENG Chuanxiang, YANG Jia, WANG Chao, ZHENG Zhi, LI Xiaopeng, DONG Di, CHANG Mengxia, ZHUANG Zhiheng et al ., [11] has counted experimental study to extract the information of GPC (ground control points) coordinated from the UAV images, to extract YOLOV5-OBB is used in combination with optimal ranking and confidence threshold filtering, data set is collected from a drone DJI Phantom 4 Pro. which attained average precision of 0.832. they also stated that the proposed methodology is good in detecting the GCP coordinates.

Xiaohe Li and Jianping Wu et al ., [24] has proposed improved YOLOV5 by including

image-adaptive enhancement which detect the vehicles under any environment, and they also came up with a tracking algorithm SORT ++ which is good at detecting the vehicle motion from the prior detections they included training images from different environmental conditions like sunny, night, snow, rainy. They stated that the results have high accuracy.

Huseyin Kusetogullari, Amir Yavariabdi, Johan Hall, Niklas Lavesson et al., [22] have suggested a new architecture for detecting and identifying handwritten numbers in nineteenth-century historical texts. The dataset is made up of 100,000 papers divided into three subsets, and the data is labeled in red, blue, and green color spaces. DIGITNET is the designed architecture, which has two architectures: DIGITNET-dect and DIGITNET-rec. They claimed that their method surpassed others in handwritten digit recognition.

Huseyin Kusetogullari,Amir Yavariabdi, Abbas Cheddad, Hakan Grahn, Johan Hall et al., [21] have created a new data set ARDIS which is taken from the Swedish church records that include historical handwritten numbers. When tested against existing datasets, it puts conventional machine learning algorithms to the test. Convolutional neural networks trained on MNIST and USPS perform best, revealing the uniqueness of ARDIS. ARDIS, which has been shared with the research community, intends to develop handwritten digit recognition algorithms.

Chapter 4

Methodology

This thesis uses literature review and experimentation as the methodologies to answer the research questions, where RQ1 is answered by a literature review which is tabulated in Section 4.1 and RQ2 is answered by experimentation which is divided into 4 parts in Section 4.2.1 of data collection describes how the data is collected, and the number of images provided by the company OVAKO. In Section 4.2.2 existing OCRs are used directly on the images, in Section 4.2.3 methodology used for old images is discussed and in Section 4.2.4 methodology used for new images is discussed.

4.1 Literature Review

This section gives detailed information regarding the motivation behind selecting algorithms used.

Table 4.1 considers six research publications found in "Google Scholar" to pick the algorithm for the current investigation. According to the research article [13], YOLO is one of the best object detection and in further research.paper [15], [27], [34], [35] different versions of the YOLO are compared and it is concluded that YOLOV5 excels in detecting objects in different cases, so YOLOV5 is finally used in detecting objects, and for oriented bounding boxes [11] YOLOV5-OBB is used, which has a high mAP.

S.No	Research work	Conclusion Drawn
1.	Understanding of object detection based on CNN family and YOLO [13]	This paper by Juan Du explores different object detection algorithms with CNN and its versions from 2012 Faster R-CNN and YOLO. a comparison is made between them in which faster R-CNN achieved MAP of 76.4 but FPS is 5 to 18, but YOLO resulted in 78.6 mAP but a greater improvement in the FPS of 155. Furthermore, the author stated that YOLOV2 has an excellent balance between speed and accuracy.

S.No	Research work	Conclusion Drawn
2	Performance Validation of YOLO variants for Object Detection [27]	This paper by Kaiyue Liu, conducts experiments on different versions of YOLO (YOLOV3, YOLOV4, and version of YOLOV5) and compares the performance of the selected algorithms. Results of the experiments indicate YOLOV4 achieved greater mAP than YOLOV3 but the processing speed was slow, but when YOLOV5 is compared to YOLOV4, YOLOV5 outperformed in both speed and mAP.
3	Comparison of Object Detection Algorithms for Street-level Objects [34]	This paper by Martinus Grady Naftali, conducts experiments in identifying street-level objects like cars, buses, etc. compares the SSD MobileNetv2 FPN-lite 320X320, YOLOV3, YOLOV4, YOLOV5l, and YOLOV5s and research concludes that YOLOV5l, MobileNetv2 FPN-lite, and YOLOV5s are promising choices for achieving the necessary balance between accuracy and speed in real-time street-level object detection tasks, including self-driving cars.
4	Comparing YOLOV3, YOLOV4, and YOLOV5 for Autonomous Landing Spot Detecting in Faulty UAVs [35]	This paper by Upesh Nepal, conducts experiments in identifying the safe landing spots in UAVs and compared three different object detection algorithms YOLOV3, YOLOV4, and YOLOV5l these algorithms are trained on the large aerial dataset, these project concludes that YOLOV5l outperforms YOLOV3 and YOLOV4 in both mAP and execution speed.
5	Tinier-YOLO: A Real-Time Object Detection Method for Constrained Environment [15]	This paper by Wei Fang et al, introduces the Tinier-YOLO which is a further advancement of Tiny-YOLO. Tinier YOLO achieved competitive mAP scores on COCO and PASCAL VOC datasets when compared to MobileNet SSD and SqueezeNet SSD, authors also stated that it achieves a smaller model size than Tiny-YOLO.

S.No	Research work	Conclusion Drawn
6	Automatic detection of aerial survey ground control points based on YOLOV5-OBB [11]	This paper introduces a solution for extracting ground control points using YOLOV5-OBB. YOLOV5-OBB is combined with ranking algorithms which produced an excellent mAP of 0.832 and a remarkable AP of 0.982 for detecting ground control points.

4.2 Experimentation

4.2.1 Data Collection

- The images of the steel bars are taken in the Rolling area and the text on the steel bars is printed there with a needle in the production unit of the OVAKO in Hofors, those images are used for the experiment and to answer RQ2 and RQ3.
- The cameras are fixed in the production unit and the cameras will capture one end of the steel bars after the needle printing, but the bars always have the marked side towards the camera.
- OVAKO has already provided 1000 images of steel bars of size 1920 X1080 from previous years and 30% of these have already been labeled by OVAKO, these 1000 images will be stated as "old images" in further sections.

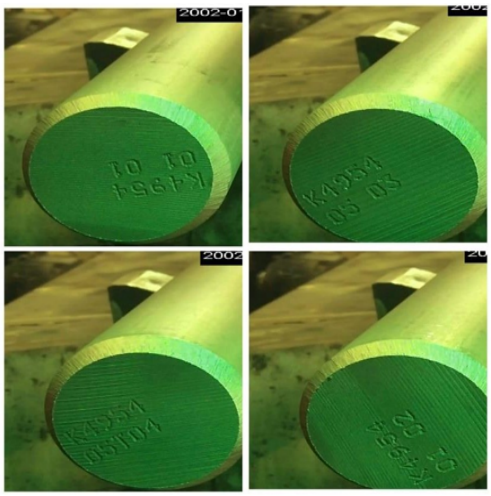

Figure 4.1: old Images

Figure 4.1 Old images are recorded in a rolling area by focusing the green light on the steel bar. The camera takes the picture, and the characters are needle-

printed on each steel bar. Each steel bar has characters such as (batch number and bar number).

- The camera system has been repaired. Ovako has experimented with the settings of both the marker and the camera during the collection, and around 3000 images. These 3000 images will be stated as “new images”.

Figure 4.2: New Images

Figure 4.2 New images are captured in a rolling area by focusing the white light on the steel bar. The camera takes the picture, and the characters are needle-printed on each steel bar and are provided with new font layouts. Each steel bar has characters such as (batch number, bar number, and checksum characters.)

4.2.2 Applying existing OCR on old images

4.2.2.1 Easy OCR

Easy OCR is a common OCR engine that can be quickly installed using the pip command in the coding interface. Once installed, all dependencies and needs are automatically downloaded, allowing users to rapidly and easily integrate OCR capabilities into their own applications.

The "Reader" tool in Easy OCR is used to read the text in multiple languages. To extract text from scanned steel bar images, the "readtext" function is used. When arguments are passed to the "Reader" method, Easy OCR also enables multi-language recognition.

Easy OCR achieved 0 accuracy when tested on a steel bar image, indicating that it was unable to extract any characters or digits on the steel bar. It is important to remember, however, that Easy OCR may not be appropriate for all OCR tasks.

4.2.2.2 Tesseract OCR

The old images are subjected to Tesseract OCR for character recognition. However, none of the characters were discovered in the first attempt, which was a disappointing result. Hyperparameter tuning was used to improve the performance of the OCR by adjusting parameters such as –psm and –oem.
The –psm argument provides 14 values ranging from 0 to 13, which determine how the script and page should be assumed. The –oem argument provides four values ranging from 0 to 3, corresponding to various OCR architectures. Various configuration combinations were explored throughout the training phase, but none of them yielded good results.

One of the main reasons for this was that the characters on the steel bar were not prominent or had a positive angle(which means the character or digits are upon the surface) like normal graphics such as number plates or door number plates. Rather, the Characters were written into the surface and had a negative angle(which means the character or digits are carved into the surface), making correct detection difficult.

4.2.3 Old images

To detect the target character, the annotated images are first subjected to YOLOV5.Then different techniques are applied for orientation and the images are submitted to character-level YOLOV5. These steps are covered in greater detail in the further chapter. The flowchart of the methodology used for old images is shown in Figure 4.3.

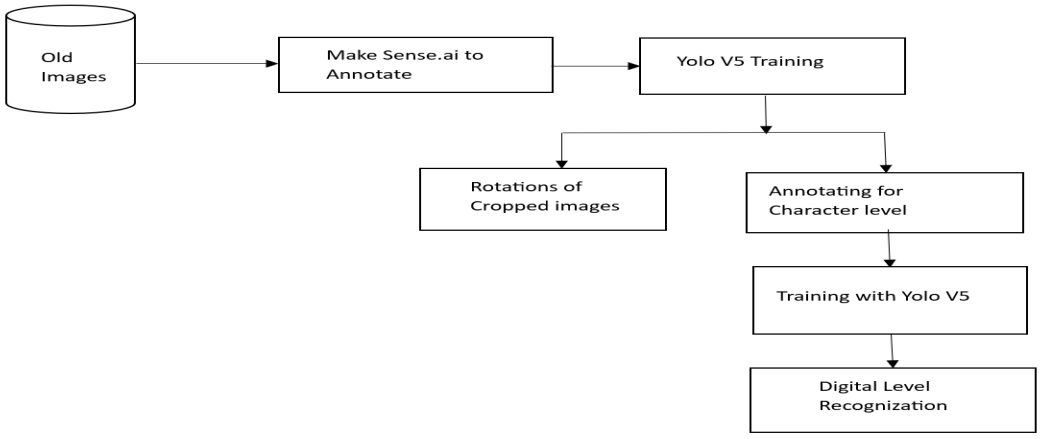

Figure 4.3: Methodology followed for old images

Figure 4.3depicts the planned methodology for old images, in which old images are first annotated with rectangular annotations in Subsubsection 4.2.3.1, then trained with YOLOV5 to obtain cropped images, as explained in Subsubsection 4.2.3.2, and finally cropped images are rotated, as explained in Section 4.2.3.3. Cropped images are then annotated directly for the character level YOLO described in Subsubsection 4.2.3.4, and characters are identified.

4.2.3.1 Rectangular bounding box Annotations

As stated in Figure 4.3 the old images were manually annotated using by make-sense.ai. The Old images are uploaded to the annotation tool for manual annotation using rectangular annotations and then annotations are downloaded in YOLOV5 format.

850 of the 1000 images are annotated, with a single class name assigned to each. These images have been divided into training and validation sets, with the remaining unannotated images left aside for testing.

Total no of Old Images	Training	Validation	Testing
1000	700	150	150
100%	70%	15%	15%

Table 4.2: Train-validation-test spliting

4.2.3.2 Training images with YOLOV5

As stated in Figure 4.3 after annotating the images those images are given to YOLOv5 training. The images and annotations are arranged in the YOLOV5 access path format, as shown in Figure 4.4, to train the images using YOLOV5. The labels folder holds the annotations of both train and validation images individually and the Images folder holds the images of both train and validation in separate folders.

Figure 4.4: Access path of YOLO

The annotations (text files) must be stored under the labels folder because YOLOV5 searches for this label folder during the training process. In addition to the images and annotations, a YAML file is generated for the dataset. This YAML file contains the paths of the train, valid, and test images, as well as the number of classes and the class names.

The YOLOV5 algorithm is clonable in our working environment. Images, labels, and YAML files are all inputs to the algorithm. The YOLOV5 method is learned using the YOLOV5s (small) weights, which were trained on the COCO dataset.

YOLOV5's training phase involves a number of trials designed to improve the model's performance. Changing hyperparameters, such as the number of epochs and the size of the training images, is one technique to accomplish this. These studies involve training the model with various hyperparameter values and recording the performance metrics that emerge.

As part of the training, experiments are also carried out to determine how the performance of the model is impacted by the size of the images. The model was trained on several image sizes in this instance, and it was discovered that even the default image size produced good mAP. In order to investigate the relationship between mAP and the number of epochs, the image size is set to 640X640 and the number of epochs is changed while running the experiment. In chapter 5 the findings are discussed.

YOLOV5 is trained using the train.py Python file, which accepts a yaml file as input and trains the algorithms using YOLOV5s weights and the input of the number of epochs.

The results of the train and detect operations are saved in two subfolders under the "run" folder, which is produced by YOLOV5. Using the "train.py" Python program, the "train" subfolder is created throughout the training process. The weights, anticipated and plotted train and validation images, graphic representations of object loss, box loss, precision, and recall for all epochs are all kept in the "train" subdirectory. Additionally, it keeps track of all the runs during the training period.

The Image with bounding boxes from the test data is kept in the "detect" folder, which is created when the "detect.py" file is used to predict the target object using the best weights from the training data. YOLOV5 offers the "save crop" and "save txt" options to extract cropped images and store the bounding box text files of the images, respectively.

The highest mAP weights are kept and utilized to find the target object in the test data. Results are provided for each epoch in YOLOV5, including object loss, box loss, precision, recall, F1 scores, and mAP [0.5], mAP [0.95]. Following the completion of all the epochs, YOLOV5 provides a summary report on the precision, recall, and mAP values as well as details on the execution time.

In the results chapter, the outcomes of the train and detect images using bounding boxes are discussed.

4.2.3.3 Rotational bounding box to correct orientation

As stated in Subsection 1.1.1, no two steel bar images, not even cropped ones, have the same orientation. The cropped images from the YOLOV5 do not have the correct orientation, so various approaches, like Tesseract, Hough transformation, and Open CV, are being tested to fix the orientation of the cropped images in accordance with the text on them.

Tesseract OCR

Despite being an algorithm for optical character recognition, it has the ability to recognize orientation and rotate the image at the identified angle. It makes use of the "imutils.rotate_bound" option, which rotates the image at that angle. However, it fails to properly orient the bounding box.

Hough Transform

It is a type of computer vision pre-processing method that locates lines in images and calculates the angle between the detected lines and the x or y-axis lines. Based on that angle, the orientation of the lines is adjusted. However, the detection bounding boxes are rectangular, always parallel to the x and y axes, and the detection of 0 degrees produced no orientation.

Open CV

In the open CV library, the minimal area of a rectangle is calculated, and the angle is then determined. The rotation matrix receives this determined angle as input and applies it to the image to rotate it.

As mentioned in Figure 4.2.3 cropped images are tried to rotate according to the text on them but none of the methods resulted good, so the result of this step is not used in the current study so YOLOV5 at the character level is applied for character extraction from steel bars.

4.2.3.4 Character level recognition of YOLOV5

As indicated in Section 4.2.2, identifying characters on steel bars is a difficult OCR problem since the characters are indented into the steel bar's surface and have a negative angle. A different strategy is taken into consideration, such as training the YOLOV5 at the character level.

As stated in Figure 4.3 the dataset is manually annotated for each character on the steel bar. To annotate makesense.ai tool is used, where on each steel bar there are 10-13 different characters that need to be annotated individually each annotated character is considered as an individual class. In total 1510 annotations were made for the 151 steel bar images, the data is divided in the 70:15:15 ratio of training, validation, and testing respectively.

However, during the training phase, it was noticed that the bounding boxes were colliding between the characters on the steel bars, which led to the loss of some of the characters. Different experiments are carried out with different sizes of data that are graphically plotted and observe the gradual increase in mAP.

According to this thesis report, character detection without maintaining a count of the characters on the steel bar led to erroneous detections and redundant detection, and in certain photos, only a small subset of characters were not detected [29].

The method used for old images provides more information about the behavior of the images and the problems with the currently utilized widespread methods. When utilized for character level recognition, YOLOV5 has problems with overlapping bounding boxes despite having demonstrated good results in finding regions of interest and extracting the suitable bounding box for them from the old images. This might disturb other characters in the picture.

These problems with character recognition prompted the development of a brand-new approach for reading characters on steel bars called YOLOV5-obb, which can read characters even when they are rotated at an angle. It is appropriate since YOLOV5-obb can recognize orientated bounding boxes. For new Images, this method is used, and it is discussed in the later parts of the Method Chapter.

4.2.4 New Images

As discussed above YOLOV5-obb object detection algorithm is used for new images. YOLOV5-obb is a bit different from YOLOV5 in acceptance of annotation and requirements and this was discussed in the further sections. Figure 4.5 is the flow chart of the methodology used for new images.

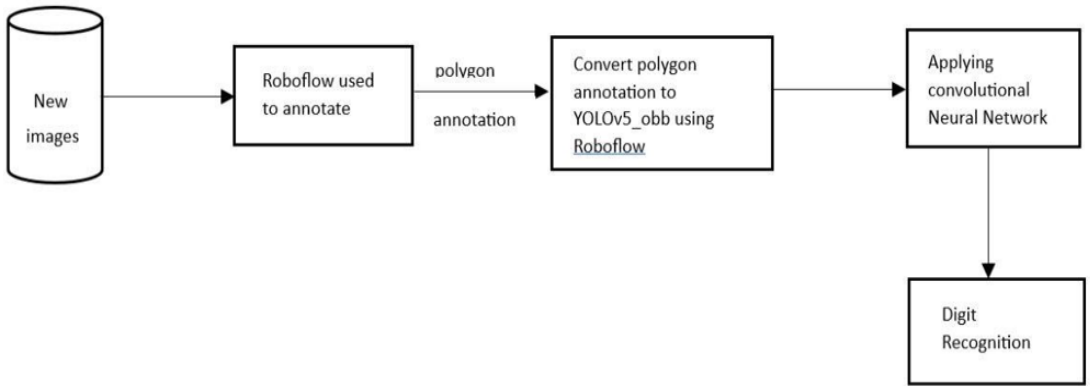

Figure 4.5: Methodology used for YOLOV5-obb

Figure 4.5 depicts the planned methodology for new images, in which new images are first annotated with polygon annotations which are explained in Subsection 4.2.4.1, then trained with YOLOV5-obb to obtain the oriented bounding boxes which are explained in Subsection 4.2.4.2, and finally a convolution neural network is applied to the previous output to remember the characters and achieve digit recognition.

4.2.4.1 Oriented bounding box Annotations:

As stated in Figure 4.5the Roboflow annotating tool is being used to annotate new images for YOLOV5-obb. Before beginning to annotate, a dataset is created by giving the number of classes and class names. The images are annotated using the polygon annotation style, which is afterward converted to YOLOV5-obb annotation

using the Roboflow tool.
Roboflow includes options for storing and tracking annotations, which aid in the generation of dataset versions when multiple training, validation, and test splits are employed. The dataset and annotations can be downloaded together, or an API key can be generated for obtaining the dataset directly in the code.

In total, 2,500 new images have been manually annotated for YOLOV5-obb using the Roboflow annotating tool. All 2500 images are under a single class. 2300 images are under training 200 images are under validation and the remaining 500 images are under testing.

4.2.4.2 Training of YOLOV5-obb

As stated in Figure 4.5 all of the Images and annotations are organized in the YOLOV5-obb access path by storing the image annotations in the labletxt folder. A yaml file is generated that contains information about the number of classes and paths to train, validate, and test images.

The YOLOV5-obb is a YOLOV5 addon that can generate orientated bounding boxes based on the angle of rotation. It can be cloned into the working interface from the "hukaixuan" git repository. The mAP metric is used to assess the model's performance. Initially, the model's baseline performance is assessed without any hyperparameter tuning or modifications to the internal training settings. However, this resulted in numerous bounding box detections for a single class and a minimal mAP. Hyperparameter tuning is used to increase the model's performance and avoid the identification of numerous bounding boxes. This entails changing the default settings of internal parameters to achieve the desired result, as detailed in the Discussion chapter.

The number of training images is gradually increased from 200 to 1000 in increments of 100 during the experiment, and then by 400 for each successive experiment until 2300 images are attained. The mAP and best weights are saved for each trial, and the outcomes of each epoch, including precision, recall, theta, box_loss, object_loss, and mAP, are recorded and analyzed in subsequent chapters. YOLOV5-obb also creates a "runs" folder in which the training and detection results for each experiment are saved.

Chapter 5

Discussions

5.1 Analysis of Research Question

In discussions, the chapter begins with the research question and then explains the method used to achieve it, findings from the research and difficulties faced during different phase is also discussed.

RQ1 What are the best-performing object detection algorithms in detecting characters on steel bars?

In answering RQ1, the literature review is used as stated in Section 4.1 for discovering optimal object identification algorithms, initially from the literature review different object detection algorithms are compared such as YOLO, Faster R-CNN, and MobileNet SSD. so YOLO is selected for the current study but in YOLO there are different versions available, again literature review is conducted in comparing different versions of YOLO. Notably, YOLOV5 was frequently favored in this analysis, with greater mAP, faster speeds, and higher FPS rates. These benefits were extended across multiple image categories, confirming YOLOV5's adaptability and making it the obvious choice for the research's object detection needs.

RQ2 How good is the proposed model at recognizing the characters on the steel bars?

The research questions is addressed utilizing image data from the company "OVAKO" which provided 4000 Images, 1000 of which are old images from prior years and 3000 of which are new images from the current year. The font style, size, and camera angle of the old and new images vary slightly. The chosen methodology is to use the YOLOV5 algorithm to locate the area of interest, pre-process the photos, and then recognize the characters with Tesseract OCR. However, because the photos are not labeled, the Makesense.ai application is utilized to annotate them.

Initially, an attempt is made to recognize the text on the old images using easy OCR and Tesseract OCR, as indicated in Section 4.2.2, by modifying the setups, however, this yielded 0 accuracy, underlining the need for a new technique. The fundamental reason for this was that the characters on the steel bar were not prominent and had a negative angle, making accurate detection difficult. Unlike traditional pictures like number plates or door number plates, the characters were carved on

the surface of the steel, making detection more difficult. As mentioned in Subsection 1.1.1, this industrial image has distinct qualities, and it is necessary to determine the position of the text in order to distinguish it from the background. This emphasizes the importance of using a specialized OCR technique that is suited to the exact properties of the image. As a result, a new technique is required, which includes first finding text, similar to object detection, and then detecting text on it. "YOLOV5" is used.

The YOLOV5 algorithm, as stated in Subsection 4.2.3.1, requires labeled data, however, no pre-annotated dataset is adequate for this industrial data. As a result, the data is manually annotated using the Makesense.ai tool. The algorithm was trained using the original parameters, but the mAP was poor. To improve mAP, the number of epochs was gradually increased to 150, and the batch size was set to 16, while training and validation remained constant. Table 5.1 shows the relation between the no. of epochs and mAP.

S.no	Name	Number of epochs	Training mAP
1	Baseline performance	3	0.30
2	HyperParameter tunning	50	0.56
3	Hyper parameter tunning	100	0.87
4	Hyperparameter tunning	150	0.995

Table 5.1: Relation between no.of epochs and mAP

The mAP increases as the number of epochs increases, and the best results are obtained with 150 epochs, 16 batch size, 700 images in training, and 150 for validation. On the validation dataset, the mAP is 0.995. Attached is a picture of validation prediction labels, which means that the algorithms used some images from validation data to predict the labels based on the weights.

Figure 5.1: Validation prediction bounding boxes of old images

To detect the bounding boxes for the test data, the best weights from the previous trial (good mAP) are saved. The YOLOV5 algorithm has a save-crop option for saving cropped photos. This option is used to obtain cropped photos for the full dataset for both the train and valid images, Figure 5.2 is the reference provided.

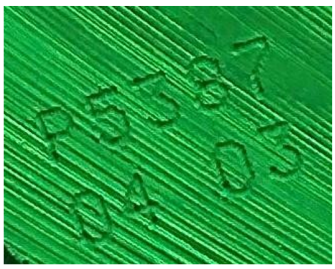

Figure 5.2: Cropped Images

An experiment is carried out in order to rotate the cropped Image to the proper orientation. Section 4.2.3.3 discusses the process used to rotate the images. To rotate the images, first determine the angle, and then rotate the images according to that angle. Tesseract OCR is used to calculate the angle of rotation, which is then utilized to rotate the images.

When the cropped image was fed into Tesseract OCR, it recognized the orientation angle as 180 and the writing as Arabic. This implies that the Tesseract OCR was unable to detect the right orientation because the cropped images are rectangular in shape, and the text on all of the cropped images must be detected in order to obtain the angle.

When the cropped image was fed into the Tesseract OCR, it recognized the orientation angle as 180 and the writing as Arabic. This implies that the Tesseract OCR was unable to detect the right orientation because the cropped images are rectangular in shape, and the text on all of the cropped images must be detected to obtain the angle.

In Section 4.2.3.4, the annotations and training details for the YOLOV5 at the character level are discussed. Multiple experiments are carried out to gradually increase the amount of training data in order to enhance the model's accuracy. Table 5.2 shows the relations between mAP and training data, as well as graph 5.3 indicating the tendency for improvements.

S.no	Size of Training data	mAP
1	22	0.17
2	66	0.24
3	111	0.37
4	130	0.54

Table 5.2: mAP and training data relation

In Figure 5.3, it can be observed that there are no signs of diminishing returns, based on the graph upon increasing the training data, mAP is also increased.

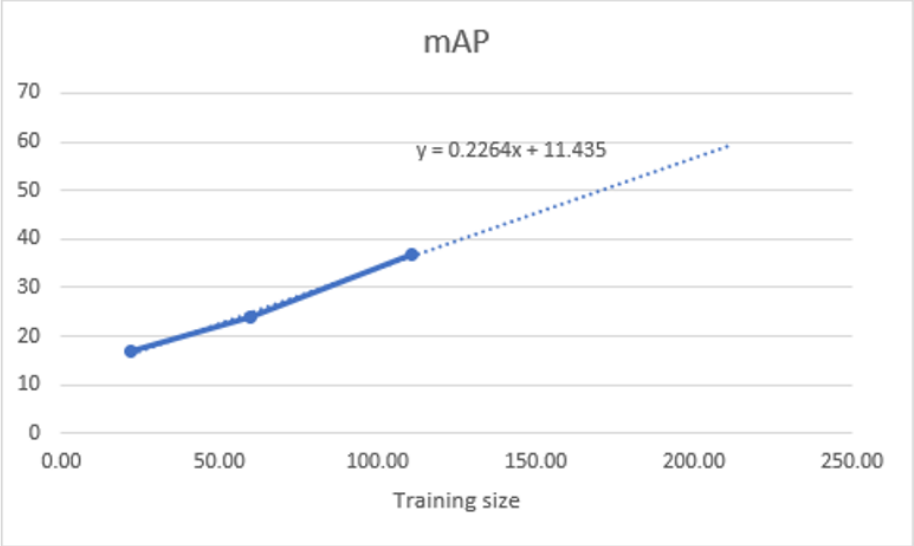

Figure 5.3: character level YOLO graph

However, annotating more than or equal to 10 characters for each image proved time-consuming, and they also found challenges due to changes in text orientation within images. Figures 5.4 and 5.5 depict the worst and best-case situations for annotating the images per character level. The change in text orientation on the images caused issues such as bounding boxes colliding with other characters or missing out parts of the character, as shown in Figure 5.4 worst-case scenario image.

Figure 5.4: Worst case scenario

Figure 5.4 depicts an example of a difficult image in a dataset that was trained with 9 classes. Some characters, such as "k", "4", "1", "5", and "0", are not fully caught within the bounding box when utilizing rectangular bounding boxes to annotate them. Furthermore, the bounding box for "0" includes a minor piece of the adjacent character "2". Object detection algorithms may struggle to effectively recognize and classify these characters as a result of incomplete bounding boxes. Improving the quality and completeness of the annotations improves the model's accuracy.

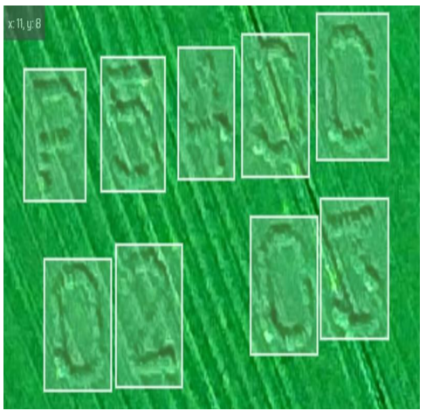

Figure 5.5: Best case scenario

Figure 5.5 depicts the best-case scenario, in which rectangular bounding boxes are employed and there are no bounding box conflicts or character exclusion concerns. The key difference between the worst-case and best-case scenario, seen in Figures 5.4 and 5.5, is the text orientation. In the best-case situation, text orientation resulted in a more precise recording of the characters' exact shapes.

To summarise, while increasing the amount of training data can enhance the model's mAP, the quality and accuracy of the annotations are critical for the model's performance on test data. Advanced annotation approaches, such as polygonal or mask annotations, or image rotation with precise orientation, can help to address the issues encountered during annotation and training.

However, as described in Section 4.2.3.3, rotation of cropped images is not successful since it requires identifying the text in the image to rotate, but the text detecting OCR method is also not recognizing the text in the image due to the negative angle of text. The YOLOV5-obb approach is used to obtain suitable orientation, and it is applied to the new images.

In Section 4.2.4 of the methodology chapter discusses the use of YOLOV5-obb on New Images, the methodology utilized, and the annotation process. Every step of YOLOV5-obb must be run with a cuda backend. Many connectivity troubles were encountered because the interface utilized was the free edition of Google Colab. Due to the free edition, the YOLOV5-obb experiment could only be done five times. Also, as the amount of data increased, the time required to run the experiment increased, resulting in the disconnection of Google Colab.

The initial performance of YOLOV5-obb was analyzed in Section 4.2.4.2, but the findings revealed a very low mAP and several bounding boxes for the same class which can be seen in Figure 5.6.

To fix this, the internal parameters of the python files train.py and general.py were changed. To obtain a single bounding box for each class, the non_max_supression_obb method's default settings for agnostics, multiclass, and nc were adjusted. The experiment was carried out with 300 training images with 32 batch sizes and no hy-

Figure 5.6: Multiple Bounding box

perparameters The results after adjusting the internal parameters to obtain a single bounding box for each class are shown in Figure 5.7.

Figure 5.7: Single bounding box

Figure 5.7 outputs a single bounding box, however the bounding box is not orientated and does not fit the target object appropriately. To accomplish this, hyperparameters such as initial learning rate and momentum were altered, and the experiment was carried out by increasing the number of images by 100 until 1000 images were obtained. The mAP improved, and the training data bounding boxes fit nicely.
Additional hyperparameters such as final learning rate, anchor, and shear are tweaked to enhance the mAP further, and the experiment is repeated every 400 images until 2300 images are reached. The mAP was 0.93, and the best weights were kept and applied to the 165 photos in testing. The experiments are listed in Table 5.3.

When just 200 images were utilized for training, it turned out that the algorithm was not accurately fitting the bounding boxes to the objects in the images, as seen in Figure 5.8. To solve this issue, the training data was expanded, and in Experiment 12, a significant improvement in performance was found, as shown in Figure 5.9. This implies that increasing the training data can increase the algorithm's predic-

Experiment Number	Training data	mAP
1	200	0.47
2	300	0.42
3	400	0.548
4	500	0.544
5	600	0.599
6	700	0.72
7	800	0.66
8	900	0.74
9	1000	0.79
10	1400	0.83
11	1800	0.87
12	2300	0.93

Table 5.3: YOLOV5-obb experiment

tion accuracy.

In experiments 1 and 2, the algorithm failed to identify the ROI nevertheless, when the data size increased in subsequent experiments, the model began to recognize the ROI. Figure 5.8 is from experiment 5 when the data size is around 600, and Figure 5.9 is from experiment 12.

Observing Figures 5.8 and 5.9, it is clear that in Figure 5.8, the ROI is recognized but it is not the right fit because it includes the excess area from the background, in Figure 5.9, the ROI is properly identified and there is an exact fit to it.

Figure 5.8: Test Image from experiment 5

It is important to note that the performance of the algorithm can be influenced by various factors, such as the quality and diversity of the training data, hyperparameter tuning, and the choice of model architecture.

Figure 5.9: Test Image from experiment 12

- **RQ3** What are the uncertainty conditions in the images that must be overcome to make better predictions?

After doing the experiment and viewing the outcomes and image conditions, the research question can be answered.

1. **Camera Focus and Positioning:** special attention be given to the location of the camera when photographing the steel bars, as the focus of the camera fluctuates, resulting in blurry and defocused shots, affecting the quality of data.

2. **Needle Care:** Careful handling of the needle used to print characters on steel bars is essential. Over time, the needle can lose its sharpness, leading to missing characters in images. Regular maintenance and replacement of the needle are recommended.

3. **Adjusting Lighting:**It is preferable to shift the positions of the light that is reflected on the steel bar while taking the picture, some of the pictures have a high focus of light, while others have a low focus, making it difficult for the system to process.

5.2 Challenges faced

1. **Difficulties Recognising Text on Steel Bars:** The first difficulty was recognizing text on steel bars inside the images. The characters were unnoticed, and their negative angle made recognition impossible.

2. **Unsuitability of existing OCR approaches:** It was discovered that existing OCR approaches, such as Easy OCR and Tesseract OCR, were unsuited for industrial image data. These approaches were unable to identify the characters on the steel bar.

3. **Lack of Labelled Data:** no pre-annotated datasets are available that match the industrial images, all the images are manually annotated.

4. **Complexity of Training Data:** Due to the complexity of the images and differences in text orientation, annotating the training data at the character level has issues.

5. **Time-Consuming Annotation:** Annotating several characters in each image took time, especially when aiming for precision in bounding boxes.

6. **Accurate Character identification:** Ensuring accurate character identification while avoiding issues such as bounding box collisions or missed characters created challenges, particularly in complex images during character level YOLOV5.

7. **Text direction Issues:** Determining and correcting the direction of text within cropped photos proved difficult. Tesseract OCR has difficulty detecting the correct orientation.

8. **Connectivity Issues in Experimentation**: When doing experiments with Google Colab, connectivity issues were found, resulting in disruptions and limitations in experimentation.

9. **Hyperparameter Tuning:** Finding the best hyperparameter values for the algorithm needed a lot of trial and error while using Tesseract OCR for character identification.

10. **Achieving a Single Bounding Box for Each Class:** The YOLOV5-obb approach requires changes to the internal parameters and settings to ensure a single bounding box for each class.

11. **Scaling Training Data:** Increasing the size of the training dataset presented issues, as it resulted in longer training durations and significant resource constraints in YOLOV5-obb.

.

Chapter 6

Results and Analysis

The Results chapter presents a thorough analysis of the literature review, the existing OCR models, manual annotations, YOLOV5, and YOLOV5-obb object detection algorithms. It includes details on the performance of the OCR models, annotation details, training images and test images, evaluation metrics, and quantitative mAP obtained for each method. The chapter compares the strengths and weaknesses of the approaches, addresses limitations, and concludes with key findings and recommendations.

6.1 Literature review results

The Table6.1 displays the results of the literature review conducted in the method chapter 4. The cite column in the Table6.1 gives the citation of the respective paper, the second column "Author" gives the name of the first author, the third column "Algorithms used" gives the algorithms used by the authors in that particular paper, and the fourth column gives a detailed explanation and criteria on which algorithm is selected. These papers were chosen based on the following criteria: "object detection, different versions of YOLO, unique version of YOLO for different problems."

Cite	Author	Algorithms used	Selected algorithm for current study and Reason
[13]	Juan Du	Faster R-CNN, YOLO, YOLOV2	YOLO was selected because when compared to other objection detection algorithms it proved its efficiency in detecting objects.
[27]	Kaiyue Liu	YOLOV3, YOLOV4, YOLOV5	YOLOV5 was selected because it excelled in detecting objects when compared to other versions.

[34]	Martinus Grady Naftali	SSD MobileNetv2 FPN-lite 320X320, YOLOV3, YOLOV4, YOLOV5l, and YOLOV5s	YOLOV5s weights are used in the current study because they proved excellent detection with more FPS
[35]	Upesh Nepal	YOLOV3, YOLOV4, and YOLOV5	YOLOV5 were selected from this is the 2nd supporting paper that when compared to remaining versions of the YOLOv5 proved its efficiency in identifying object s under difficult background.
[15]	Wei Fang	Tinier YOLO	Even though this version of YOLO was not chosen, this study article provides an alternative perspective on employing YOLO, demonstrating that not just normal versions but also these kinds of unique versions can be employed depending on the problem.
[11]	chuan xiang	YOLOV5-OBB	YOLOV5-OBB

Table 6.1: Literature review Results

Model Name	Version used	Character recognization Accuracy
Tesseract OCR	5.0	0
Easy OCR	1.6.2	0

Table 6.2: existing OCR Character recognition Accuracy

6.2 Experimentation

6.2.1 Existing pre-trained OCR models

The most recent versions of Tesseract ocr and Easy OCR were applied to industrial images, but they produced 0% accuracy. Even after altering the parameters, no existing OCR recognized a single character. This result might be attributable to a variety of factors, including image quality, complicated backgrounds, and the angle of the characters printed on the steel bars.

6.2.2 YOLOV5 Results

The following subsections contain the results of Old images which include Annotating images, testing, and training with YOLOV5.

6.2.2.1 Manual annotations

The "old images" manual annotation comprises of rectangular bounding boxes saved in text files. Each annotation file is associated with a single image and contains five number values per line. The first value reflects the class, with 0 signifying that all objects have the same class. The remaining four numbers represent the x-coordinate, y-coordinate, height, and width of the bounding box, which define the ROI. This annotation style allows for precise ROI localization and identification.

facilitating tasks such as object identification algorithm evaluation, outcome comparison, or machine learning model training. Overall, hand annotations are critical for correct labeling and analysis of "old images".

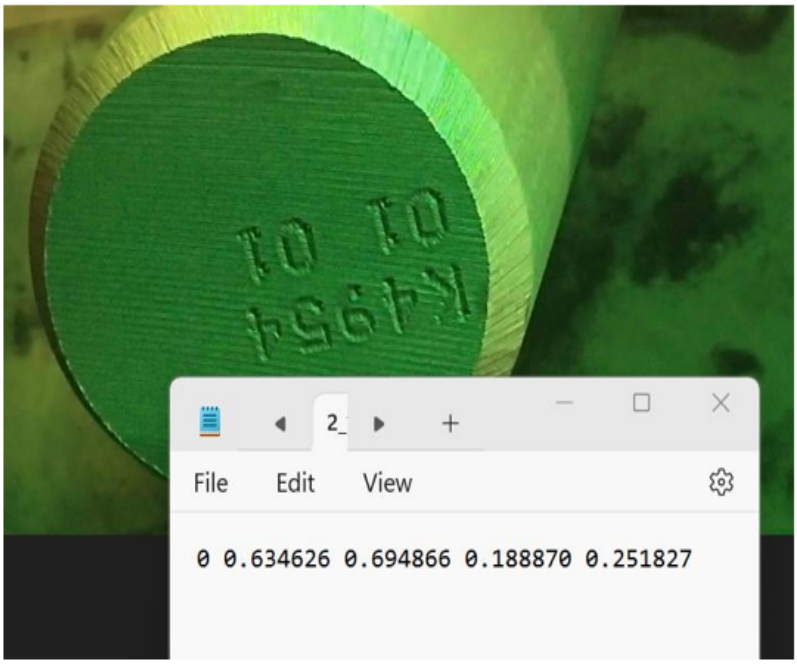

Figure 6.1: Manual Annotations of YOLOV5

6.2.2.2 Train Images

As mentioned method chapter, 70% of the data was set aside for model training, with a batch size of 16. The training phase produced an outstanding model performance, with a mAP score of 0.99. Figure 6.2 shows the batch 0 training photos, indicating that the model correctly identified the class in each image. This exceptional result illustrates the model's high accuracy and efficacy in recognizing the class across the training dataset.

The achieved mAP score of 0.99 and accurate class identification in all training images highlight the model's robustness and ability to generalize successfully to unseen data. These findings give the model confidence in its capacity to make reliable predictions because it has effectively learned the distinctive traits of the defined class. This outstanding performance lays a solid platform for the model's subsequent evaluation and application in real-world scenarios.

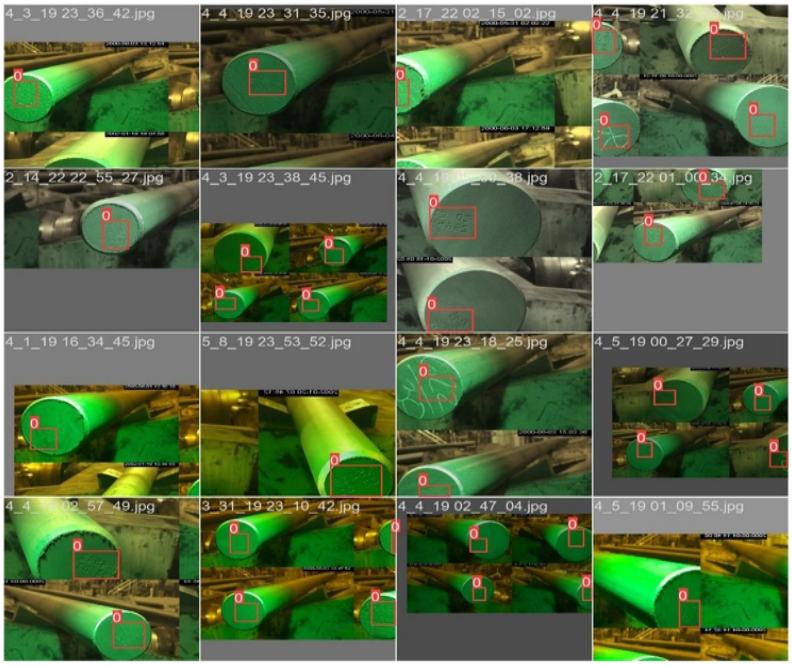

Figure 6.2: Training Batch of YOLOV5

6.2.2.3 Test Images

The next stage was to examine the test data using the model weights that had been learned with a remarkable 0.99 mAP on the training data. The goal was to use the trained model to construct bounding boxes that correctly identified the regions of interest (ROI) in the test images. Bounding boxes were created by applying the model to the test dataset, indicating the positions and dimensions of the ROIs. These bounding boxes function as visual representations, allowing for the identification and retrieval of relevant data inside the areas of interest. Figure 6.3 is the test data that detected the ROI with 0.95 mAP.

Figure 6.3: Test Image of YOLOV5

6.2.3 YOLOV5 at Character level

These sections hold the results of YOLOV5 applied at the character level along with the Annotations of character level YOLO and training the model.

6.2.3.1 Manual annotations

The manual Annotations for the cropped version of old images identify the characters, and the text files contain the bounding box details of each character along with the class it belongs to.

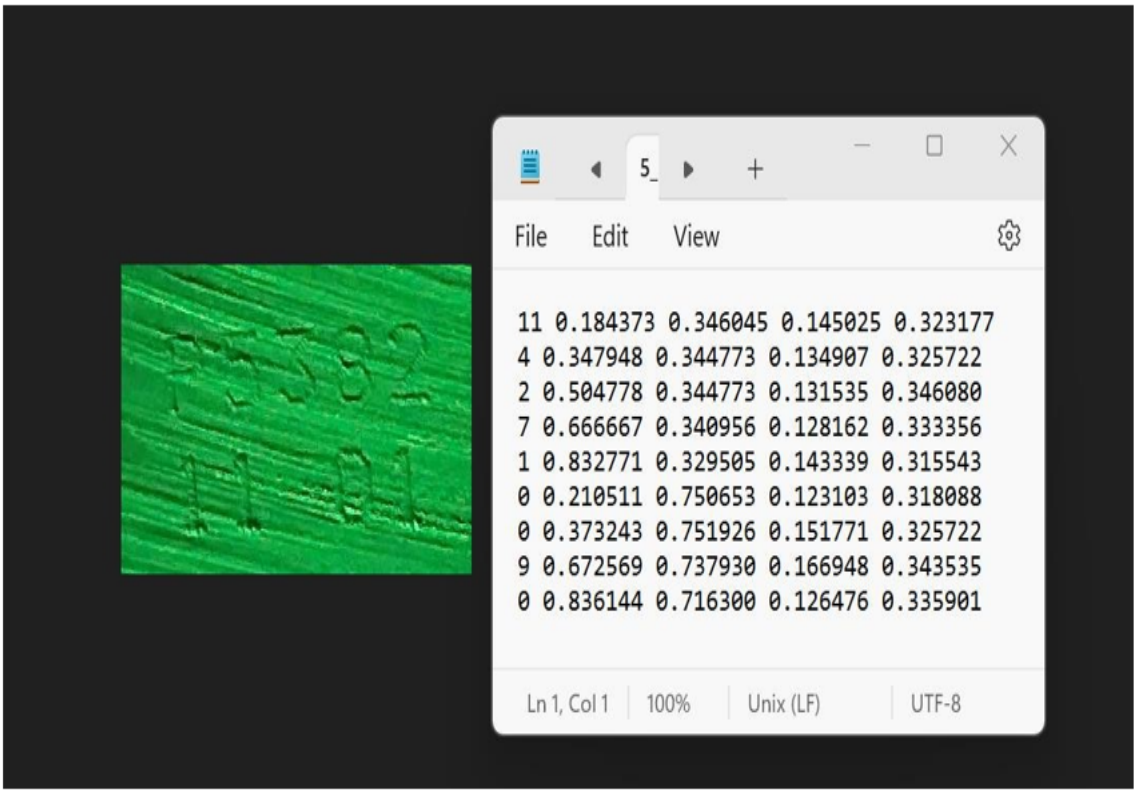

Figure 6.4: Annotation at Character level

6.2.3.2 Train Images

Figure 6.5 depicts the training of YOLOV5 at the character level, with a batch size of 8. The model was trained on a dataset of 130 manually annotated photos and yielded a mAP of 0.54. Characters were detected and marked with their associated class names in each image in the dataset.

The obtained mAP score of 0.54 demonstrates the model's ability to recognize and classify characters in training images. This performance illustrates how well it's able to handle character-level object detection. The highlighted class names associated with each character reveal more about the model's comprehension of distinct character classes and its ability to distinguish between them.

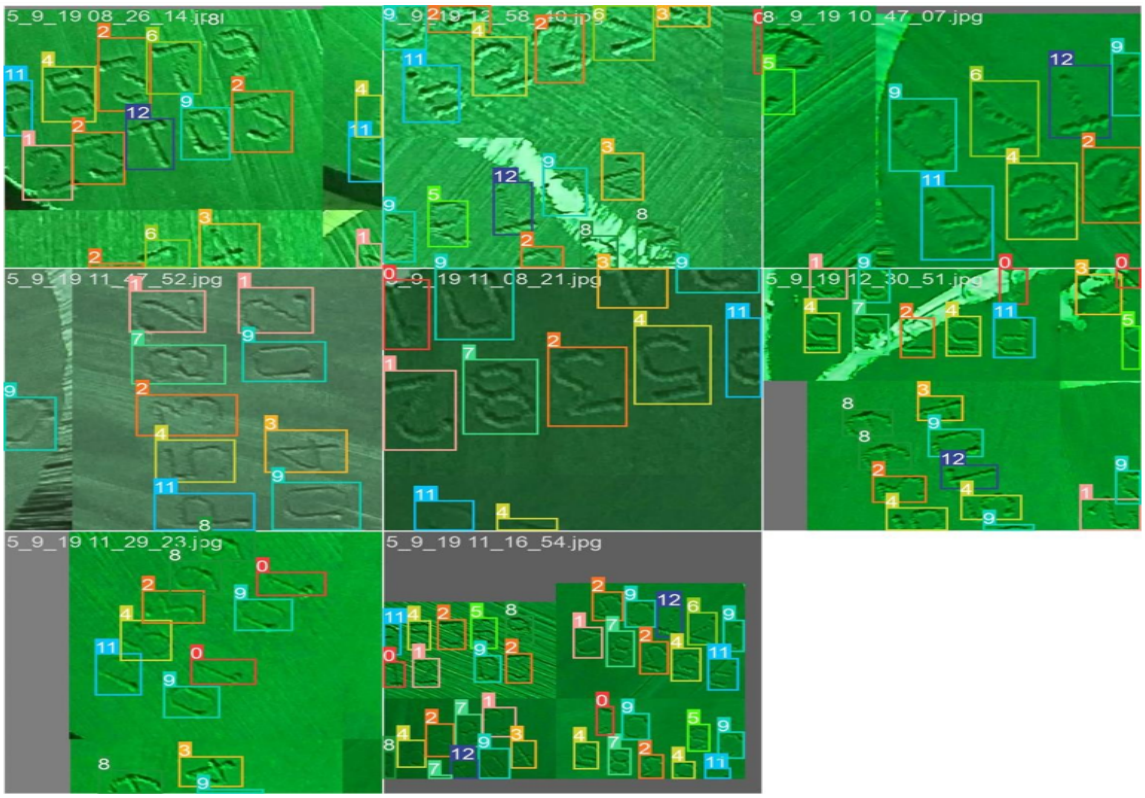

Figure 6.5: Training YOLO at Character level

6.2.4 YOLOV5-obb Results

This section contains the findings of the new images that used YOLOV5-obb to recognize the target object, as well as manual annotation for YOLOV5-obb and a discussion of the testing and training images and their outcomes.

6.2.4.1 Manual annotations

Figure 6.6 depicts the manual polygon annotation of the YOLOV5obb. the text files contain 11 values, the first of which is the class name and the rest are the YOLOV5-obb bounding box values used to identify the ROI.

The number of data points in the text file utilized to detect the ROI may vary based on the complexity and form of the annotated item as the new images follow polygon annotation.

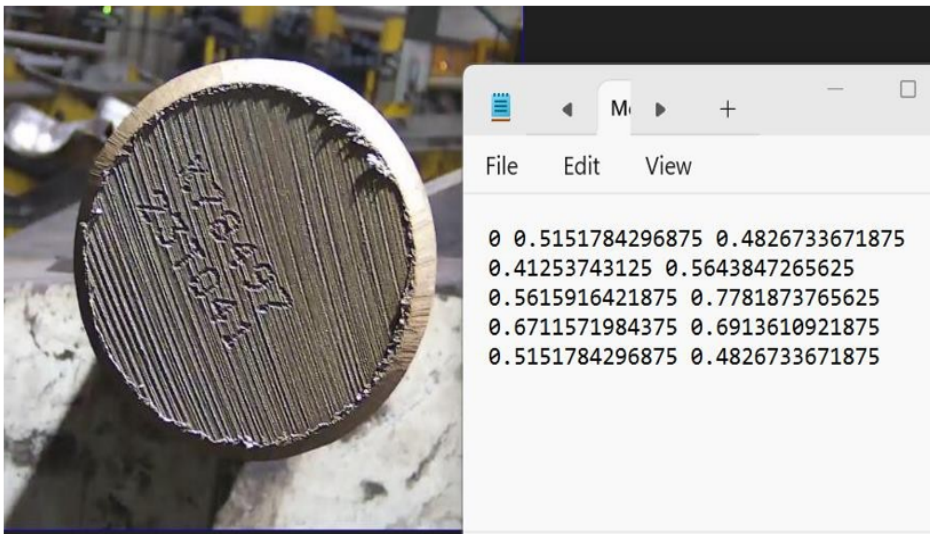

Figure 6.6: Annotation of YOLOV5-obb

6.2.4.2 Train Images

Figure 6.7 is the training picture of YOLOV5-obb with 2300 images and achieved a 0.93 mAP, utilizing the batch size of 4. The ROI is properly identified and there is a proper orientation in the bounding box.

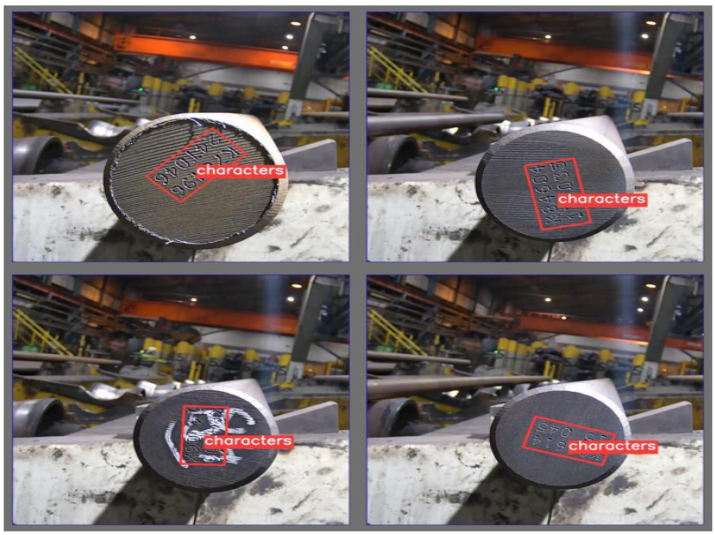

Figure 6.7: Training Batch 0 at YOLOV5-obb

Figure 6.7 shows four images from the training set, each with a polygon bounding box that fits exactly to the item.

When comparing Figures 6.3 and 6.7, it is obvious that in Figure 6.3, excess background is captured along with the target object and is rectangular in shape, despite the target object being turned at an angle. However, in Figure 6.7, the bounding box fits exactly to the target object, and the bounding boxes are also orientated in accordance with the target object's orientation.

6.2.4.3 Test Images

Figure 6.8 is a test image that is obtained by using the best weights of the YOLOV5-obb. Even though the test mAP is 0.93, false detections are observed which are mentioned in the discussions chapter.

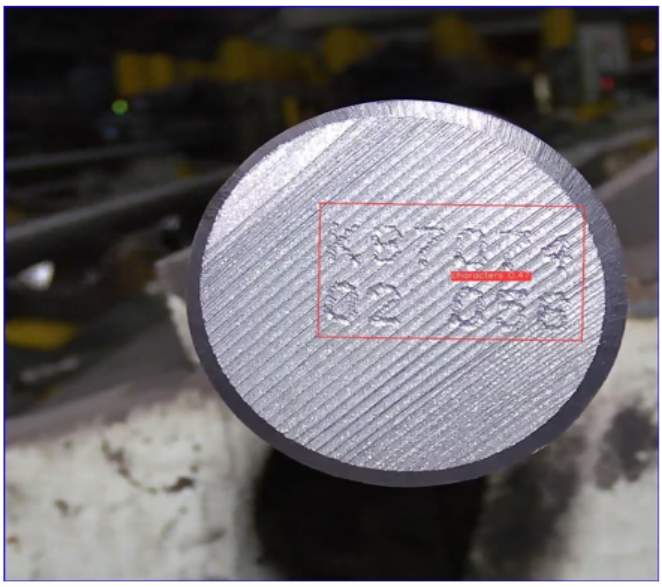

Figure 6.8: Testing image of YOLOV5-obb

6.3 Compartive Analysis of the proposed model with another model

According to the results shown in Table 6.3, the proposed model YOLOV5 excels in detecting ROI regardless of the background conditions, especially when compared to other models using the same dataset from the same company.

Model	test mAP
ROI:SSD MobileNet V2 FPNLite 320x320 [29]	90.4
YOLOV5(proposed model)	95.0

Table 6.3: Comparitive results

The results indicate that YOLOV5 achieves a good average precision of 95% in correctly identifying ROI even in greater disturbances in the backgrounds.

6.4 Validity Threats

Potential challenges to the validity of this thesis are thoroughly addressed in this section, which includes both internal and external validation. The study's robustness in terms of object detection is highlighted by addressing these threats. This methodology ensures a thorough evaluation of the research's reliability and application.

6.4.1 Internal Validity

Internal validity in the context of the YOLOV5 object detection problem refers to the extent to which the observable outcomes, results, and conclusions obtained from the experiment are a true depiction of the actual causal links and effects inside the system under study [5].

In this industrial problem, the object detection algorithm performed well in detecting the target object even in difficult situations with blurry backgrounds and disturbing backgrounds, and some of the steel bars are not completely in the picture even in that case YOLOV5 showed promising results by detecting the target object.

6.4.2 External Validity

The extent to which the findings and conclusions gained from the experiment can be generalized beyond the specific dataset, settings, and conditions of the study is referred to as external validity in the context of YOLOV5 object detection [5].

In general, YOLOV5 is good at detecting objects however it depends on several factors like background, and environmental conditions like lighting, orientation, and image quality There are many different scenarios in real-world applications so it cannot be expected that YOLOV5 will perform well for all the datasets

6.4.3 Conclusion Validity

In conclusion of Validity, the output of the algorithms is validated in each stage with an mAP score and through visualization.

Chapter 7

Conclusions and Future Work

7.1 Conclusion

In conclusion, this thesis addressed the research objectives systematically. It began with a literature analysis to pick relevant algorithms, ultimately settling on YOLOV5. Thanks to OVAKO for providing a dataset with 4,000 images in total, resulting in 1,000 old images and 3,000 new images of steel bars.

The main challenge was to create an OCR system that is as accurate as a human eye reader. Initial OCR attempts on antique photos produced low accuracy. Shifting to YOLOV5 for region recognition yielded good results, but text orientation remained a challenge. YOLOV5 had limited success at the character level.

To overcome these issues, YOLOV5-obb was employed for new image analysis. Evaluation metrics, including accuracy and mAP, revealed that existing OCR had 0% accuracy, while YOLOV5 achieved an mAP of 0.95 on test images. YOLOV5-obb achieved an mAP of 0.93, especially in text orientation detection.

Due to time constraints, the final research segment was partially implemented. In essence, this research harnessed YOLOV5 and YOLOV5-obb to identify steel bar characters. It delineated regions, extracted images, and performed character-level identification, yielding promising results. Future studies should further explore and refine these approaches. This research contributes valuable insights into industrial image character recognition, emphasizing the ongoing need for advancements in object detection and OCR techniques.

Due to time constraints, the final research phase was only partially implemented. Essentially, this study used YOLOV5 and YOLOV5-obb and made an attempt to identify the character on the steel bar. It identified regions, extracted images, and performed character-level identification, and the results were promising. Future research should investigate and improve on these approaches. This study adds to our understanding of industrial image character recognition by emphasizing the continual need for breakthroughs in object detection and OCR approaches.

7.2 Future Work

- Training the YOLOV5-obb with some more data to increase the test data mAP, so that exact oriented bounding boxes can be seen for the test data.

- As mentioned in the conclusion the cropped images from the YOLOV5-obb and then they are subjected to a convolution neural network to get the accurate character.
- Some improvements in the image quality like lighting on the image, and the position of the camera, the distance between the camera and the steel bar, will definitely have an impact on the performance

References

[1] [Online]. Available: https://roboflow.com/model/yolov5-obb

[2] T. Ahmad, Y. Ma, M. Yahya, B. Ahmad, S. Nazir, and A. u. Haq, "Object detection through modified yolo neural network," *Scientific Programming*, vol. 2020, pp. 1–10, 2020.

[3] I. H. Al Amin, F. H. Arby *et al.*, "Implementation of yolo-v5 for a real time social distancing detection," *Journal of Applied Informatics and Computing*, vol. 6, no. 1, pp. 01–06, 2022.

[4] Y. Amit, P. Felzenszwalb, and R. Girshick, "Object detection," *Computer Vision: A Reference Guide*, pp. 1–9, 2020.

[5] C. Andrade, "Internal, external, and ecological validity in research design, conduct, and evaluation," *Indian journal of psychological medicine*, vol. 40, no. 5, pp. 498–499, 2018.

[6] R. Arief, A. B. Mutiara, T. M. Kusuma *et al.*, "Automated hierarchical classification of scanned documents using convolutional neural network and regular expression." *International Journal of Electrical & Computer Engineering (2088-8708)*, vol. 12, no. 1, 2022.

[7] R. Brown, "3d cuboid annotation for self-driving cars and robots," Jul 2023. [Online]. Available: https://medium.com/cogitotech/3d-cuboid-annotation-for-self-driving-cars-and-robots-a84c624fa77f

[8] M. Busta, L. Neumann, and J. Matas, "Fastext: Efficient unconstrained scene text detector," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1206–1214.

[9] M. Chablani, "Yolo - you only look once, real time object detection explained," Jun 2023. [Online]. Available: https://towardsdatascience.com/yolo-you-only-look-once-real-time-object-detection-explained-492dc9230006

[10] Y.-C. Chiu, C.-Y. Tsai, M.-D. Ruan, G.-Y. Shen, and T.-T. Lee, "Mobilenet-ssdv2: An improved object detection model for embedded systems," in *2020 International conference on system science and engineering (ICSSE)*. IEEE, 2020, pp. 1–5.

[11] C. Chuanxiang, Y. Jia, W. Chao, Z. Zhi, L. Xiaopeng, D. Di, C. Mengxia, and Z. Zhiheng, "Automatic detection of aerial survey ground control points based on yolov5-obb," *arXiv preprint arXiv:2303.03041*, 2023.

[12] F. Dai, B. Chen, H. Xu, Y. Ma, X. Li, B. Feng, P. Yuan, C. Yan, and Q. Zhao, "Unbiased iou for spherical image object detection," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 1, 2022, pp. 508–515.

[13] J. Du, "Understanding of object detection based on cnn family and yolo," in *Journal of Physics: Conference Series*, vol. 1004. IOP Publishing, 2018, p. 012029.

[14] I. El Naqa and M. J. Murphy, *What is machine learning?* Springer, 2015.

[15] W. Fang, L. Wang, and P. Ren, "Tinier-yolo: A real-time object detection method for constrained environments," *IEEE Access*, vol. 8, pp. 1935–1944, 2020.

[16] J. Han, Y. Liao, J. Zhang, S. Wang, and S. Li, "Target fusion detection of lidar and camera based on the improved yolo algorithm," *Mathematics*, vol. 6, no. 10, p. 213, 2018.

[17] R. Huang, J. Pedoeem, and C. Chen, "Yolo-lite: a real-time object detection algorithm optimized for non-gpu computers," in *2018 IEEE international conference on big data (big data)*. IEEE, 2018, pp. 2503–2510.

[18] C. Jeeva, T. Porselvi, B. Krithika, R. Shreya, G. S. Priyaa, and K. Sivasankari, "Intelligent image text reader using easy ocr, nrclex & nltk," in *2022 International Conference on Power, Energy, Control and Transmission Systems (ICPECTS)*. IEEE, 2022, pp. 1–6.

[19] P. Jiang, D. Ergu, F. Liu, Y. Cai, and B. Ma, "A review of yolo algorithm developments," *Procedia Computer Science*, vol. 199, pp. 1066–1073, 2022.

[20] R. Kundu, "Yolo algorithm for object detection explained [+ examples]," 2023.

[21] H. Kusetogullari, A. Yavariabdi, A. Cheddad, H. Grahn, and J. Hall, "Ardis: a swedish historical handwritten digit dataset," *Neural Computing and Applications*, vol. 32, no. 21, pp. 16 505–16 518, 2020.

[22] H. Kusetogullari, A. Yavariabdi, J. Hall, and N. Lavesson, "Digitnet: A deep handwritten digit detection and recognition method using a new historical handwritten digit dataset," *Big Data Research*, vol. 23, p. 100182, 2021.

[23] R. Laroca, E. Severo, L. A. Zanlorensi, L. S. Oliveira, G. R. Gonçalves, W. R. Schwartz, and D. Menotti, "A robust real-time automatic license plate recognition based on the yolo detector," in *2018 international joint conference on neural networks (ijcnn)*. IEEE, 2018, pp. 1–10.

[24] X. Li and J. Wu, "Extracting high-precision vehicle motion data from unmanned aerial vehicle video captured under various weather conditions," *Remote Sensing*, vol. 14, no. 21, p. 5513, 2022.

[25] X. Li, Z. Cai, and X. Zhao, "Oriented-yolov5: A real-time oriented detector based on yolov5," in *2022 7th International Conference on Computer and Communication Systems (ICCCS)*. IEEE, 2022, pp. 216–222.

[26] J.-P. Lin and M.-T. Sun, "A yolo-based traffic counting system," in *2018 Conference on Technologies and Applications of Artificial Intelligence (TAAI)*. IEEE, 2018, pp. 82–85.

[27] K. Liu, H. Tang, S. He, Q. Yu, Y. Xiong, and N. Wang, "Performance validation of yolo variants for object detection," in *Proceedings of the 2021 International Conference on bioinformatics and intelligent computing*, 2021, pp. 239–243.

[28] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*. Springer, 2016, pp. 21–37.

[29] R. Locher, "Npocr – needle printer character recognition: Deep learning-based image id recognition," Jul 2022. [Online]. Available: https://www.diva-portal.org/smash/record.jsf?pid=diva2%3A1680036&dswid=7580

[30] Makesense.ai, "Makesense.ai." [Online]. Available: https://sourceforge.net/software/product/makesense.ai/#:~:text=makesense.ai%20is%20a%20free,to%20be%20truly%20cross%2Dplatform.

[31] T. Malathi, D. Selvamuthukumaran, C. D. Chandar, V. Niranjan, and A. Swashthika, "An experimental performance analysis on robotics process automation (rpa) with open source ocr engines: Microsoft ocr and google tesseract ocr," in *IOP Conference Series: Materials Science and Engineering*, vol. 1059, no. 1. IOP Publishing, 2021, p. 012004.

[32] map, "Mean average precision (map) explained: Everything you need to know." [Online]. Available: https://www.v7labs.com/blog/mean-average-precision#:~:text=The%20mAP%20is%20calculated%20by,over%20a%20number%20of%20classes.&text=The%20mAP%20incorporates%20the%20trade,metric%20for%20most%20detection%20applications.

[33] J. Martínez Segura, "Muretools. an optical music recognition supporting system," 2021.

[34] M. G. Naftali, J. S. Sulistyawan, and K. Julian, "Comparison of object detection algorithms for street-level objects," *arXiv preprint arXiv:2208.11315*, 2022.

[35] U. Nepal and H. Eslamiat, "Comparing yolov3, yolov4 and yolov5 for autonomous landing spot detection in faulty uavs," *Sensors*, vol. 22, no. 2, p. 464, 2022.

[36] C. Patel, A. Patel, and D. Patel, "Optical character recognition by open source ocr tool tesseract: A case study," *International Journal of Computer Applications*, vol. 55, no. 10, pp. 50–56, 2012.

[37] S. Pokhrel, "Image data labelling and annotation: Everything you need to know," *Towards Data Science*, 2020.

[38] r. Roboflow, "Roboflow," Jul 2023. [Online]. Available: https://sourceforge.net/software/product/Roboflow/

[39] M.-C. Roh and J.-y. Lee, "Refining faster-rcnn for accurate object detection," in *2017 fifteenth IAPR international conference on machine vision applications (MVA)*. IEEE, 2017, pp. 514–517.

[40] D. Sahoo and Y. Dhawan, "Evaluation of supervised and unsupervised machine learning classifiers for mac os malware detection," *Handbook of Big Data Analytics and Forensics*, pp. 159–175, 2022.

[41] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4510–4520.

[42] A. Saxe, S. Nelli, and C. Summerfield, "If deep learning is the answer, what is the question?" *Nature Reviews Neuroscience*, vol. 22, no. 1, pp. 55–67, 2021.

[43] M. R. Servedio, "The what and why of research on reinforcement," *PLoS Biology*, vol. 2, no. 12, p. e420, 2004.

[44] D. Shah, "Mean average precision (map) explained: Everything you need to know," *Retrieved November*, vol. 4, p. 2022, 2022.

[45] P. Sharma, "Basic introduction to convolutional neural network in deep learning," Aug 2023. [Online]. Available: https://www.analyticsvidhya.com/blog/2022/03/basic-introduction-to-convolutional-neural-network-in-deep-learning/

[46] R. Shwartz-Ziv, R. Balestriero, and Y. LeCun, "What do we maximize in self-supervised learning?" *arXiv preprint arXiv:2207.10081*, 2022.

[47] R. Smith *et al.*, "Tesseract ocr engine," *Lecture. Google Code. Google Inc*, 2007.

[48] A. P. Tafti, A. Baghaie, M. Assefi, H. R. Arabnia, Z. Yu, and P. Peissig, "Ocr as a service: an experimental evaluation of google docs ocr, tesseract, abbyy finereader, and transym," in *Advances in Visual Computing: 12th International Symposium, ISVC 2016, Las Vegas, NV, USA, December 12-14, 2016, Proceedings, Part I 12*. Springer, 2016, pp. 735–746.

[49] Y. Tian, G. Yang, Z. Wang, H. Wang, E. Li, and Z. Liang, "Apple detection during different growth stages in orchards using the improved yolo-v3 model," *Computers and electronics in agriculture*, vol. 157, pp. 417–426, 2019.

[50] g. h. Ultralytics, "Ultralytics/yolov5: Yolov5 in pytorch gt; onnx gt; coreml gt; tflite." [Online]. Available: https://github.com/ultralytics/yolov5

[51] H. Wang, C. Pan, X. Guo, C. Ji, and K. Deng, "From object detection to text detection and recognition: A brief evolution history of optical character recognition," *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 13, no. 5, p. e1547, 2021.

[52] J. Wu, "Introduction to convolutional neural networks," *National Key Lab for Novel Software Technology. Nanjing University. China*, vol. 5, no. 23, p. 495, 2017.

[53] H. S. Yi, "Automated approaches to enable innovative civic applications from citizen generated imagery," Ph.D. dissertation, University of South Florida, 2023.

[54] A. Younis, L. Shixin, S. Jn, and Z. Hai, "Real-time object detection using pre-trained deep learning models mobilenet-ssd," in *Proceedings of 2020 the 6th international conference on computing and data engineering*, 2020, pp. 44–48.

[55] X. Zeng, W. Ouyang, B. Yang, J. Yan, and X. Wang, "Gated bi-directional cnn for object detection," in *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part VII 14*. Springer, 2016, pp. 354–369.

Faculty of Computing, Blekinge Institute of Technology, 371 79 Karlskrona, Sweden