



A comparison of Advanced deep learning algorithms for multi-digit recognition in historical documents.

Machine Learning

Sai Teja Palla

Manaswini Chethireddy

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 to any other institution to obtain a degree.

Contact Information:

Author(s):

Sai Teja Palla

E-mail: sapb20@student.bth.se

Manaswini Chethireddy

E-mail: mach21@student.bth.se

University advisor:

Hüseyin Kusetogullari

Department of Computer Science Engineering

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. The context of this thesis is to detect the handwritten digits from ancient scriptures using different advanced deep learning algorithms to make the detection process more efficient, as historical documents are assets for future generations and should be secured properly. In this thesis, four deep learning algorithms are used to detect the digits and evaluated them using performance metrics.

Objectives. This study first considers a few deep learning algorithms for detecting the digits, considering the different challenges of the handwritten digits, and then finds the best algorithm among them using metrics to know which is the best performing deep learning model.

Methods. Literature Review and experimentation are the research strategies employed in this study. We have chosen four advanced deep-learning methods to identify handwritten digits in digit string images. Each method is trained and tested using the DIDA dataset. Performance evaluation is conducted to determine the best method based on all analyses from experiments on all selected DL methods.

Results. The augmented and digit string datasets are used for training and testing the deep learning models. The chosen models are evaluated using metrics for an efficient model. The results from the experimental evaluation show the best deep learning model among the selected models for detecting and recognizing multi-digit strings in historical handwritten digit images.

Conclusions. Results obtained from the performance metrics of the respective algorithm say that the YOLOv7 algorithm has more efficiency and accuracy compared to YOLOv5, RetinaNet, and Faster R-CNN for detecting handwritten digits in images of historical documents.

Keywords: Machine Learning, Deep Learning Methods, Handwritten digits, Digit Detection, Image processing on document images, YOLOV5, Faster R-CNN, RetinaNet, YOLOV7

Acknowledgments

We want to express our gratitude to Dr.Hüseyin Kusetogullari, our thesis supervisor, for allowing us to collaborate with him. With his ongoing support and suggestions for improvement during the research process, we were able to complete our thesis.

Finally, we would like to express gratitude to our parents and friends for their unwavering moral support during our thesis.

Contents

Abstract	i
Acknowledgments	iii
Nomenclature	xi
1 Introduction	1
1.1 Aim and objectives	2
1.2 Research questions	2
1.3 Ethical aspects	3
1.4 Outline	3
2 Background	5
2.1 Machine Learning	5
2.2 Deep Learning	6
2.2.1 Convolutional neural networks	7
2.2.2 R-CNN	7
2.2.3 The Target Detection Models	8
2.3 Deep Learning Algorithms	8
2.3.1 Faster-RCNN	9
2.3.2 RetinaNet	10
2.3.3 You Only Look Once (YOLO)	11
2.3.4 YOLOv5	13
2.3.5 YOLOv7	14
2.4 The Handwritten Digit	17
3 Related Work	19
3.1 Research Gap	21
4 Method	23
4.1 Literature Review	23
4.2 Experimentation	24
4.3 Tools and Libraries	25
4.3.1 Core Libraries	25
4.3.2 Algorithm specific libraries	26
4.4 Dataset	26
4.5 Research Design	28
4.6 Implementation	29

4.6.1	Data Preprocessing	29
4.6.2	Model Training	30
4.6.3	Model Evaluation	31
5	Results and Analysis	33
5.1	Literature Review	33
5.2	Literature Review Analysis	35
5.3	Experimental Results	36
5.3.1	Results of Faster R-CNN	36
5.3.2	Results of RetinaNet	36
5.3.3	Results of YOLOv5	36
5.3.4	Results of YOLOv7	36
5.4	Analysis of Experimental Results	41
5.4.1	Accuracy	41
5.4.2	Precision	41
5.4.3	Recall	42
5.4.4	F1 score	43
6	Discussion	45
6.1	Analysis on Detection Results	45
6.2	Threats to validity	46
6.2.1	Internal validity	46
6.2.2	External validity	46
6.2.3	Construct Validity	47
6.2.4	Conclusion validity	47
7	Conclusions and Future Work	49
7.1	Conclusion	49
7.2	Future Work	49
	References	51
A	Supplemental Information	57

List of Figures

2.1	AI ML DL Comparision [20]	6
2.2	Deep Neural Network Architecture [21]	7
2.3	Faster R-CNN Architecture [24].	10
2.4	RetinaNet Architecture [22]	11
2.5	YOLO Architecture [23]	12
2.6	YOLOv5 Architecture [26]	13
2.7	E-ELAN [25]	15
2.8	Compound Scaling [25]	15
2.9	YOLOv7 Module planned re-parameterization [25]	16
2.10	Coarse-to-fine auxiliary head in the YOLOv7 network [25]	16
2.11	YOLOv7 Label Assigner [25]	17
4.1	Fixed length multi-digit string images [10]	27
4.2	variable length multi-digit string images [10]	27
4.3	single handwritten digit samples in the 416×416 images [10]	27
4.4	Research Design	28
4.5	Digits frequency in the training data	30
5.1	Faster R-CNN detection results	37
5.2	Faster-RCNN's Confusion Matrix	37
5.3	RetinaNet detection results	38
5.4	RetinaNet's Confusion Matrix	38
5.5	YOLOv5 detection results	39
5.6	YOLOv5's Confusion Matrix	39
5.7	YOLOv7 detection results	40
5.8	YOLOv7's Confusion Matrix	40
5.9	Comparison of detection accuracy for digit string images.	41
5.10	Comparison of detection precision for digit string images.	42
5.11	Comparison of detection Recall for digit string images.	42
5.12	Comparison of detection F1 score for digit string images.	43
A.1	Annotation file for YOLOv5 and YOLOv7 in text format	57
A.2	Annotation file for Retinanet in .csv format	57
A.3	Annotation file for Faster R-CNN in .tfrecord format.	58

List of Tables

4.1	Google Colab Pro Environment	25
4.2	Details of the hyperparameter search space are displayed in the table.	31
5.1	The results of the Literature Review are shown in the table.	35
5.2	The results of digit detection are shown in the table.	41

Nomenclature

AI Artificial Intelligence

CNN Convolutional Neural Networks

CSPNet Cross Stage Partial Network

DIDA Digital Dataset

DL Deep Learning

E-ELAN Extended Efficient Layer Aggregation Network

FAIR Facebook AI Research

Faster R-CNN Faster Region-Based Convolutional Network

FPN Feature Pyramid Network

GPU Graphics Processing Unit

IOU Intersection over Union

ML Machine Learning

MNIST Modified National Institute of Standards and Technology database

NIST SD19 National Institute of Standards and Technology database

PANet Path Aggregation Network

R-CNN Region-Based Convolutional Neural Network

ROI Region of interest

RPN Region Proposal Network

SVM Support Vector Machines

USPS dataset United States Postal Service dataset

YOLOv5 You only live once version 5

YOLOv7 You only live once version 7

YOLO You only live once

Handwritten recognition is vital in information processing in this day and age of digitization. A vast amount of information is available on paper, and digital files require less processing than traditional paper files [1]. The ancient archives and documents must be kept for many years as historical evidence. The digital format of the paper is easier to preserve than the physical format, as it can be shared across multiple platforms and accessible from any location in the world [2]. But, most of the ancient archives are written by hand. Therefore, it would be more beneficial to convert important handwritten notes, records, and papers into digital text to keep and preserve them as files.

As there are numerous documents written by hand, converting handwriting to text is laborious and time-consuming because it requires examining the document and writing every word. This issue can be resolved by automating the translation of handwritten notes to digital text. This procedure is simplified by a machine learning model that recognizes text from an image of text and converts it to digital text. The handwritten digit recognition system aims to translate handwritten digits into machine-readable formats. Vehicle license plate recognition, postal paper sorting services, historical document preservation in archaeology departments, old document automation in libraries and banks, and other applications are significant applications of Handwritten detection and recognition [3]. These fields deal with enormous databases and require high identification accuracy, low computing complexity, and consistent identification system performance.

Machine learning has undergone significant growth in recent years and is widely used for data analysis and computation. [4]. One subfield of machine learning, deep learning, utilizes neural networks to achieve intelligent applications. In particular, Convolutional Neural Networks (CNNs) play a crucial role in deep learning and are widely used for image classification, object detection, and segmentation [5].

A CNN is composed of one or more layers of convolution units, which take into account the context and shared knowledge among small groups of nodes. Convolutional layers in a neural network are defined as layers where each node receives input directly from nodes in the previous layer [5]. Since the discovery of GPU usage in machine learning in 2005, the CNN sector has seen a dramatic improvement in its visual classification performance [6]. As a result, CNNs have become a key component in advanced deep-learning architectures for object detection.

Deep learning algorithms have achieved human perfection in identifying handwritten digits compared to a single typical machine learning algorithm because they consist of a wide variety of models due to the flexibility that neural networks provide when building a full-fledged end-to-end model. There are many issues in Handwritten digit detection where the digits have different font sizes, and the scripts may be old, damaged, or faded. The current standard datasets MNIST [7], NIST SD19 [8], and USPS [9] are built from a non-degraded handwritten document with clear backgrounds and modern handwritten styles. These features of the datasets have led the existing handwritten digit detection and recognition methods to result in poor performance on historical document images.

Handwritten digit detection is a vital task, and it has different challenging problems, such as touching digits, overlapping digits, bleeding through, and pale digits. These distortions make it difficult to position and recognize individual digits. We use the DIDA dataset [10], which contains the digit strings from the Swedish historical documents associated with these artifacts. There are many Swedish historical handwritten digit datasets [11] [12], but we used the DIDA dataset for our research. Existing approaches perform well with current datasets but not with historical document images. We train and verify advanced deep-learning models on historical document images and compare their performance with metric measures such as Accuracy, Precision, Recall and F1-score.

1.1 Aim and objectives

This research aims to detect the historical handwritten digit strings using advanced deep learning algorithms using the DIDA datasets. [10]

The objectives of our research are as follows,

OBJ1 : Find the effective deep learning algorithms for detecting handwritten digits in multi-digit string images.

OBJ2 : Detect the handwritten digits in multi-digit string images using advanced deep learning algorithms.

OBJ3 : Evaluate each deep learning algorithm's accuracy and efficiency for detecting and recognizing the multi-digit strings in historical document images.

1.2 Research questions

- **RQ1: What are the effective deep learning models for detecting multi-digit strings in historical handwritten document images?**

Justification : With RQ1, we intend to highlight the numerous deep-learning algorithms used in previous research for text and digit recognition and evaluate them for our comparative study.

- **RQ2: Which deep learning methods provide the best detection result for multi-digit detection?**

Justification : The objective of our thesis is to compare different deep learning algorithms and analyze which algorithm performs the best among them. We conduct Experimentation to determine the optimum algorithm for recognizing a multi-digit string. The developed recognition models are validated and compared using different performance metrics such as accuracy, precision, etc..

1.3 Ethical aspects

No standard data is manipulated in this thesis to use deep learning techniques for detecting historical handwritten digits. University policies are not broken during research, and no one's privacy is invaded. The thesis is no harm to the environment and society, and the datasets are used with consent and contain no sensitive or illegal data. All the research and study were conducted systematically without harming anyone or anything.

1.4 Outline

This paper further continues with the combination of the following chapters,

Chapter 2 is about the background of all the algorithms and terms required to understand the thesis concept.

Chapter 3 is about the previous related works of other researchers on digit detection, YOLOv5, YOLOv7, RetinaNet, Faster R-CNN, and handwritten digits. This chapter also elaborates on the research gap of earlier studies.

Chapter 4 is the main part of this paper, describing the study's methods. Literature review and Experimentation tell about how the research design is presented and which tools and datasets are taken for implementation.

Chapter 5 is all about the results and the analysis, and it details the detection results of the algorithms and metrics used to evaluate them.

Chapter 6 is filled with discussions on the study's results and threats that occurred while implementing the thesis.

Chapter 7 concludes the research with its outcomes, validating that the study's problem statement and research questions are answered. The future work of this thesis is mentioned at the end of this paper.

This chapter gives a brief explanation of the topics which should be known before the implementation of the methods. Topics related to Deep learning algorithms and their architectural implementations are elaborated one by one.

2.1 Machine Learning

Machine learning (ML) is a subset of Artificial Intelligence (AI) whose main goal is to make machines work without the need for explicit programming. AI is a science that strives to make machines able to think like humans. The process is hard to implement, but machine learning and deep learning are the ways to make it happen by making decisions with the large amount of data taken through analysis. [13] Machine Learning is a cognitive computing approach where the computer learns from the problem set and then tries to find the pattern and structure in the dataset. The machines are programmed to learn from the static input, where they learn and improve their performance without external help.

ML techniques have a vast hold in the categories like speech processing, computer vision, the Internet of Things, neuroscience, health, and natural language understanding. Machine learning requires less computing power, and there are different ways for a machine to learn [14]. They are simple and complex. Artificial neural networks are a complex way of learning. Deep learning (DL) uses artificial neural networks. Machine learning approaches are divided into four types: Supervised Learning, Semi-supervised Learning, Unsupervised Learning, and Reinforcement Learning.

- **Supervised Learning:** The input and output data are given to the machine; it has to find a method to validate the data set. The algorithm learns by observing the identified patterns in data and then makes predictions until it achieves a high level of accuracy with the help of external corrections [14]. Classification, Regression, and Forecasting are the different algorithms under supervised learning.
- **Semi-supervised learning:** Semi-supervised learning is a combination of labeled and unlabeled data where the machine learning algorithm learns to label the unlabeled data [14].
- **Unsupervised learning:** Unsupervised learning contains only inputs. The learning algorithm interprets and organizes the data and then makes the de-

cisions [14]. Clustering and dimension reduction come under unsupervised learning.

- **Reinforcement learning:** Reinforcement learning is a trial-and-error method where the machine learning algorithm explores different possibilities evaluating all results to give a final solution with the given set of rules. The algorithm learns from experience, modifies, and gives the best outcome [14].

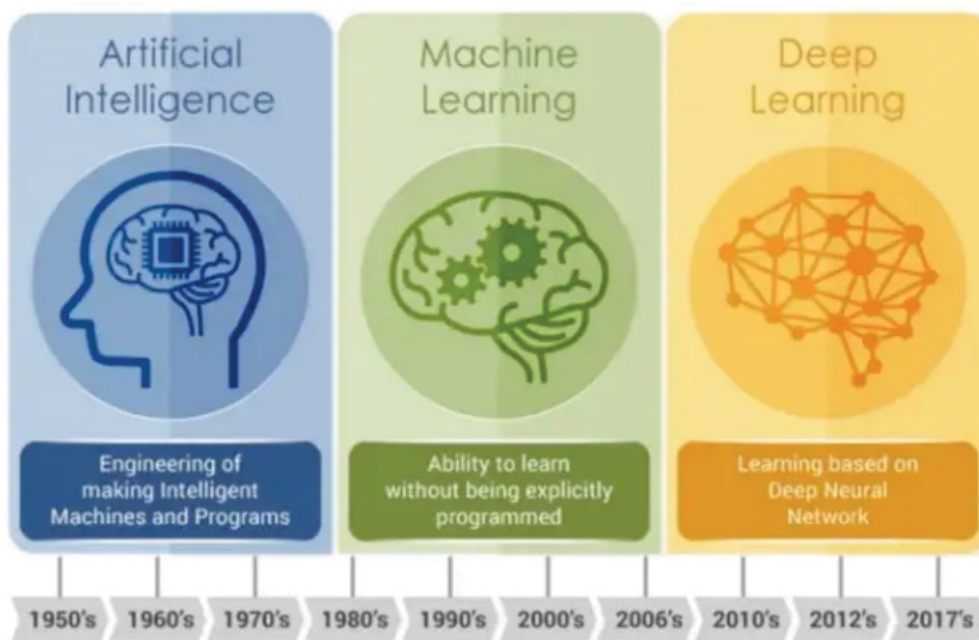


Figure 2.1: AI ML DL Comparison [20]

2.2 Deep Learning

Developers are delving into deep learning and machine learning methods to make machines more intelligent. A human learns to accomplish a task by practicing and repeating it repeatedly until they have learned the procedure by heart. Their brain's neurons then automatically fire, enabling them to quickly carry out the acquired task. This is also quite similar to deep learning. For different kinds of issues, it employs various neural network topologies. [13] All deep learning algorithms come under machine learning algorithms, but not all machine learning algorithms are deep learning algorithms.

Massive volumes of data must be fed into deep learning algorithms for the machine to produce correct findings without outside assistance. Deep learning is more effective than ML at analyzing unstructured data, videos, and photos by using artificial neural networks [15]. A neural network resembles or represents the structure of the human brain. It has synthetic neurons, or nodes, which are interconnected in three layers: the input layer, a hidden layer or layers, and the output layer, as shown in figure 2.2.

Real-time object detection is an essential task that is often a key component in Deep Learning. When performing image recognition tasks, an object detector uses

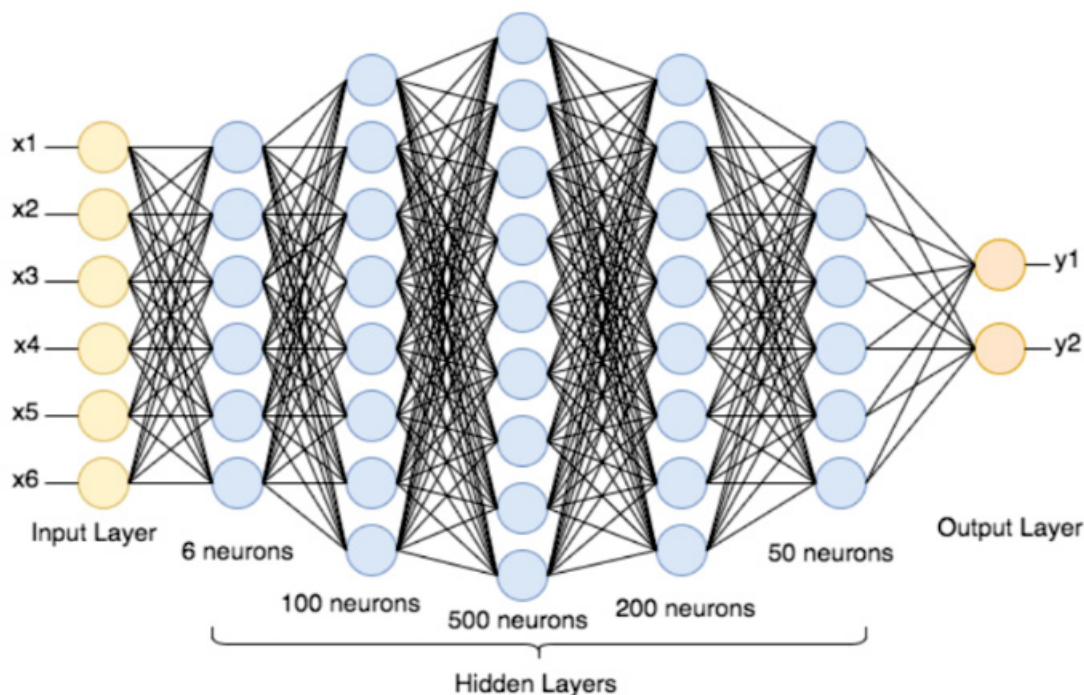


Figure 2.2: Deep Neural Network Architecture [21]

an object detection method that predicts bounding boxes and class probabilities for each object in the input image [17]. The majority of methods extract features from the image to estimate the likelihood of the learned classes using a convolutional neural network (CNN).

2.2.1 Convolutional neural networks

In deep learning, CNN is used for image classification, object detection, and segmentation. One or more layers of convolution units are present in a CNN, and they consider the context and shared knowledge in small groups. When each node receives input from nodes in the previous layer directly adjacent, it is regarded as a convolutional layer in a neural network. As a result, Convolutional neural networks are used in sophisticated deep-learning architectures for object detection. CNNs are employed because they can perform automated learning with limited manual programming. [18]

2.2.2 R-CNN

R-CNN (Regions with Convolutional Neural Network features) is an object detection system that uses deep learning to identify objects in images. The R-CNN system consists of three main components: [19]

- A selective search algorithm that generates region proposals, or potential object locations, in the input image.

- A convolutional neural network (CNN) extracts features from the region proposals and outputs a fixed-length feature vector.
- A linear support vector machine (SVM) classifier uses feature vectors to classify each region proposal into one of a set of predefined object classes.

R-CNN provided the first evidence that deep learning can be used effectively for object detection and motivated the development of more efficient object detection systems. The main limitation of R-CNN is its computational cost, as it requires running the CNN on each region proposal individually, making the system slow and infeasible for real-time object detection. Faster R-CNN is the advancement of the R-CNN algorithm, which is discussed in the other section.

2.2.3 The Target Detection Models

The algorithms used in this paper for object detection in images can be classified into two target detection deep learning algorithms. Many modern target detection models are based on deep learning, specifically convolutional neural networks (CNNs). They are,

- **Two-stage target detection algorithm:**

This algorithm is based on region recommendation. According to the regional suggestions, the algorithm generates a series of candidate boxes and then classifies and filters them [27]. R-CNN, fast R-CNN, and faster R-CNN represent the two-stage target detection. Methods that come under this category usually exhibit high detection accuracy with low detection efficiency as the extraction process is complicated.

- **One-stage target detection algorithm:**

This algorithm does not need to extract candidate regions; it is based on a regression model [16]. The main aim of the algorithm is to combine the target region prediction and target category prediction into a single convolutional neural network model to meet the requirements of real-time detection. Representative one-stage algorithms include RetinaNet and YOLO series algorithms.

2.3 Deep Learning Algorithms

Through the Literature review, which is done by referring to the relevant studies for identifying suitable algorithms, we have concluded to continue with the four deep learning algorithms. The literature review and its results are explained in the next chapters. The advanced deep learning algorithms used in this study are Faster-RCNN, RetinaNet, YOLOv5, and YOLOv7. These algorithms are explained here with their architectural implementation.

2.3.1 Faster-RCNN

A Faster-RCNN algorithm is a more advanced version of existing approaches, such as R-CNN and Fast-RCNN, that use the Edge Boxes technique to generate region proposals for potential object areas. Faster-functionality RCNNs have been altered because it now uses a region proposal network (RPN) instead of the Edge Boxes algorithm to generate region proposals directly as part of the framework as shown in figure 2.3. Therefore, faster-computational RCNN's complexity for producing region proposals is significantly lower. RPN finalizes the anchor box selection by displaying the most likely anchor boxes that contain the regions of interest. As a result, region proposal generation in Faster-RCNN is faster and more sensitive to input samples [28] [29].

2.3.1.1 Architecture

The faster R-CNN architecture is comprised of two networks:

- **Region Proposal Network (RPN):** This component generates candidate regions (also called proposals) in the input image that could potentially contain an object. The RPN uses a sliding window approach to scan the image and generates proposals based on anchor boxes, which are predefined bounding boxes of different aspect ratios and scales. The maximum amount of k-anchor boxes are created by this network. For each of the several sliding positions in the image, the default value of k=9 and three aspect ratios of 1:1, 1:2, and 2:1 are used. Therefore, we obtain $N = W * H * k$ anchor boxes for a convolution feature map of $W * H$. (Where N represents the number of parameters, W and H represent the width and height of the feature map produced by the shared feature extraction CNN, K represents the number of anchor boxes used in the RPN.) These are sent to the intermediate layers to receive the output.
- **Object Detection Network:** The object detection network utilized in Faster R-CNN and Fast R-CNN is primarily similar. In terms of a backbone network, it is likewise compatible with VGG-16. The bounding box regressor is also employed in predicting the object and its bounding box, together with twin tiers of the softmax classifier and the RoI pooling layer for creating region proposals of fixed size.

The Softmax Classifier takes the features extracted from the region proposal by the shared feature extraction CNN and uses them to calculate the probability of each class. **The RoI (Region of Interest) Pooling Layer** is a special layer in the detection network of Faster R-CNN that is used to extract features from the region proposals generated by the RPN. It takes the feature map produced by the shared feature extraction CNN and crops a fixed-size feature map for each region proposal. This helps to ensure that the feature maps extracted from the region proposals are of consistent size, regardless of the size of the region proposal in the input image.

With the use of TensorFlow, Faster R-CNN can be implemented as follows: Using the input, ConvNet gets about an image; it creates a feature map of the image. On

these feature maps, a region proposal network is used. ConvNet then returns the object proposals and their object scores. These recommendations are then subjected to an RoI pooling layer to produce a small feature map with a fixed size. The recommendations are sent to a fully connected layer that consists of a linear regression layer and a softmax layer. This procedure categorizes items and generates their bounding boxes.

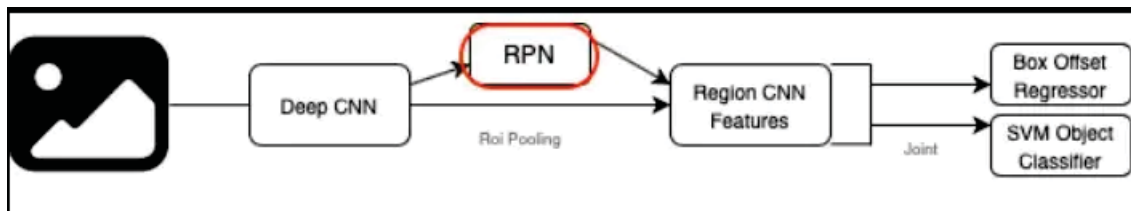


Figure 2.3: Faster R-CNN Architecture [24].

2.3.2 RetinaNet

RetinaNet is a one-stage object recognition technique that accounts for class imbalance by cutting down on the loss given to images with accurate classification. There is a class imbalance when there are more background examples than examples of the object of study. The Facebook AI Research (FAIR) team, which is now known as "Meta AI" has introduced the RetinaNet model to address the difficulty of detecting dense and small objects. [30]

It has a broad attraction of users towards it as an object detection model that can also be applied to both aerial and satellite data.

2.3.2.1 RetinaNet Architecture

An architecture for a RetinaNet model consists of the following four main parts: [31]

- a) **Bottom-up Pathway:** This backbone network, independent of the size of the input image or the backbone, calculates the feature maps at various scales as shown in a layer-(a) of figure 2.4.
- b) **Top-down pathway and lateral connections:** Top-down pathway upsamples the geographically coarser feature maps from higher pyramid levels, while lateral connections combine top-down and bottom-up layers with the same spatial size.
- c) **Classification subnetwork:** The classification subnetwork predicts the likelihood that an object will be present at each spatial location for each anchor box and item class.
- d) **Regression subnetwork:** It regresses the offset for each ground-truth object's bounding boxes relative to the anchor boxes.

Feature Pyramid Network: Image pyramids are the foundation, and feature pyramids are built on top of them. This implies that one would subsample an image into images of lower resolution and smaller size as shown in a layer-(b) of figure 2.4. Then, hand-engineered features are retrieved from each layer of the pyramid to find the items. The pyramid becomes scale-invariant as a result. But this method uses a lot of memory and computing power. Combining low-resolution semantically powerful features with high-resolution semantically weak features, FPN develops an architecture with rich semantics at all levels. This is done by building a top-down pathway that connects to bottom-up convolutional layers on the side.

Focal Loss (FL): The introduction of Focal Loss, an improvement over Cross-Entropy Loss (CE), addresses the class imbalance issue with single-stage object detection models. Due to intensive anchor box sampling, single-stage models have a severe foreground-background class imbalance problem. At each pyramid layer in RetinaNet, thousands of anchor boxes may be present.

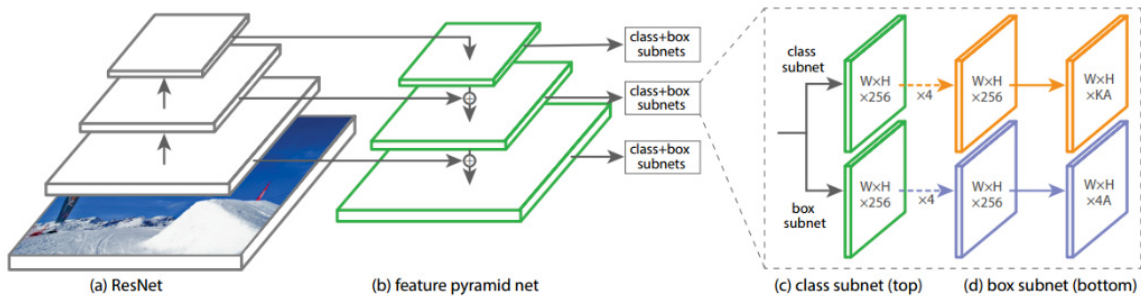


Figure 2.4: RetinaNet Architecture [22]

2.3.3 You Only Look Once (YOLO)

You Only Look Once is real-time object detection and state-of-the-art algorithm. The algorithm works as the name says; detecting objects requires a single forward propagation through a neural network. YOLO algorithm uses convolutional neural networks (CNN) to detect objects in real-time. Figure 2.5 shows the YOLO architecture with how it works with convolutional layers. The CNN in this process is used to predict various class probabilities and bounding boxes simultaneously [33].

YOLO models are object detectors that operate in a single stage. A backbone enhances image frames in a YOLO model. The Head of the network receives these properties after they have been integrated and mixed in the neck of the network, which then predicts the positions and types of objects around which bounding boxes should be drawn.

The YOLO framework has three main components-

- **Backbone:** It extracts the required features of an image and sends them to the Head through the neck.

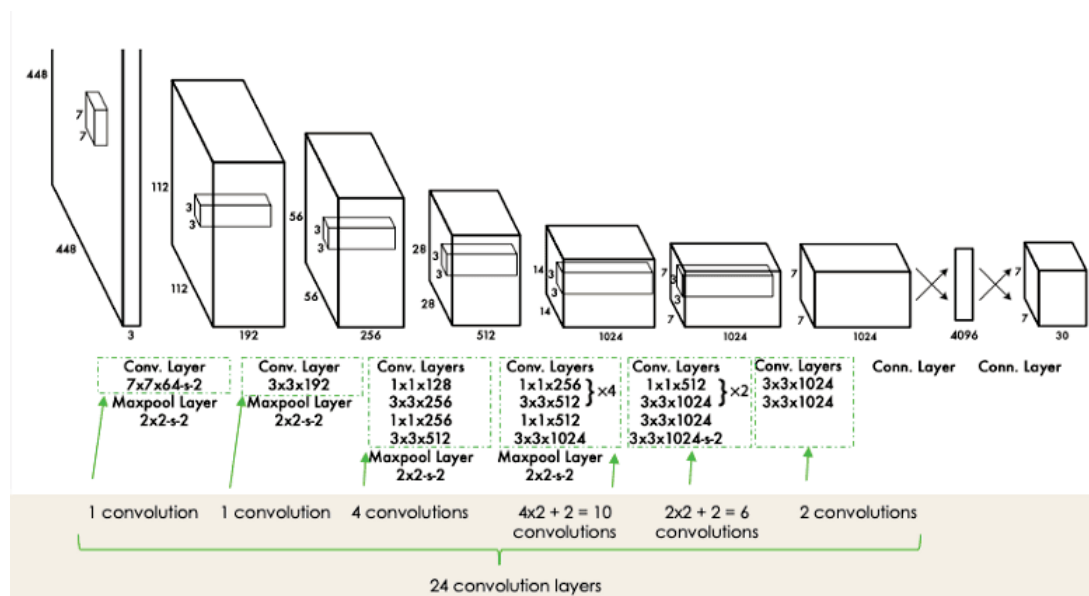


Figure 2.5: YOLO Architecture [23]

- **Neck:** It creates the feature pyramids from the feature maps extracted by the backbone.
- **Head:** It gives final output or final detection, which are in the output layers present in it.

The YOLO algorithm includes the following three methods- [32]

- **Residual blocks:** Here, the image is divided into small grids, then every grid cell acts as a detector. The grid cells detect the objects that are within them.
- **Bounding box regression:** A bounding box highlights an object with an outline in an image for object detection. YOLO uses single bounding box regression for the bounding box in the image to predict the attributes like – Width (bw), Height (bh), Class (c), Bounding box center (bx, by).
- **Intersection Over Union (IOU):** IOU describes how boxes overlap in object detection. So, YOLO uses IOU to surround the complete object with an output box. The projected bounding box will be identical to the actual bounding box if the IOU score is 1, and this method eliminates bounding boxes that are not equal to the actual bounding box.

The YOLO algorithm consists of various variants. They include YOLOv2, YOLOv3, YOLOv4, and YOLOv5, YOLOv7. In this paper, we have used YOLOv5 and YOLOv7.

2.3.4 YOLOv5

It is a model that uses PyTorch, which is simple and easy to deploy and implement. Object detection has different requirements like prediction accuracy, detection speed, and training time; YOLOv5 uses different scales on networks to meet these requirements. YOLOv5 network has high detection accuracy, lightweight characteristics, and fast detection speed simultaneously, as it is suitable for deployment to the embedded devices to implement real-time detection. [34] [35]

2.3.4.1 Architecture

Backbone: The Backbone model extracts features from an input image. A feature extractor only applies convolutional layers like kernel, stride, and batch normalization to input images to extract critical features like edges, forms, etc. In YOLO v5, a backbone of CSP (Cross Stage Partial Networks) is employed to remove valuable properties from an input image.

Neck: In addition to the Feature pyramid network, the network was created to make Multi-scale predictions in the Neck layer. The multi-scale prediction uses three different grid values to transmit images to detect objects of various sizes. Massive grid pictures identify small items, while small grid images detect large objects.

Head: The last stage of detection is primarily the responsibility of the Head. Using anchor boxes, it builds final output vectors with bounding boxes, objectness scores, and class probabilities.

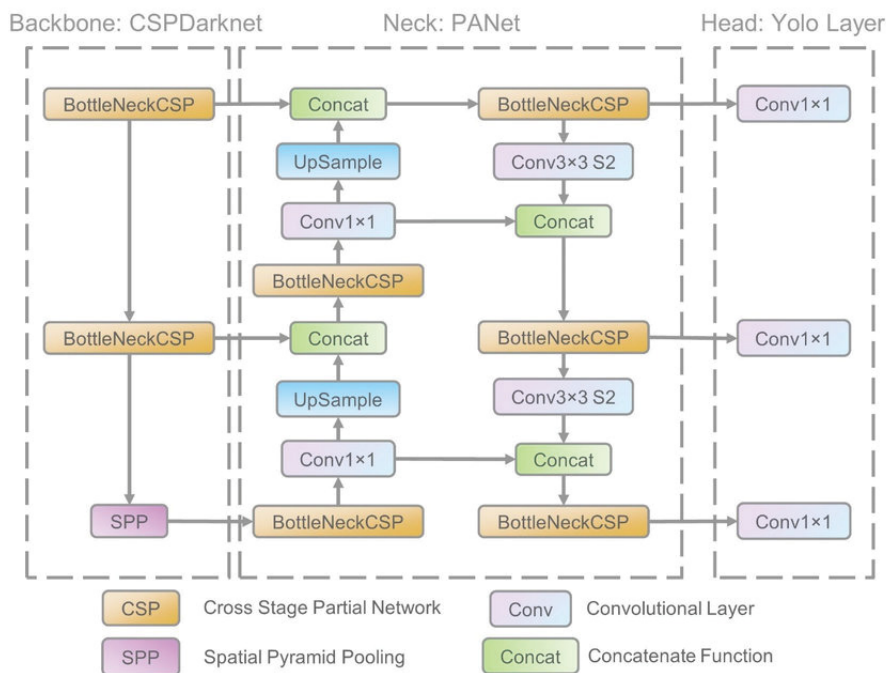


Figure 2.6: YOLOv5 Architecture [26]

The same three parts are used in all YOLOv5 models: a backbone made of CSPDarknet53, a neck made of SPP and PANet, and the Head from YOLOv4.

First, YOLOv5 builds CSPDarknet, the backbone of Darknet, by integrating a cross-stage partial network (CSPNet) [34] into it. By combining the gradient changes into the feature map and resolving the issues with repeated gradient information in large-scale backbones, CSPNet reduces the model's parameters and FLOPS (floating-point operations per second), ensuring inference speed and accuracy while also shrinking the model's size. Speed and accuracy are essential in detecting forest fires, and the compact model size also determines the efficiency of inference on edge devices with limited resources. To improve information flow, the YOLOv5 used the path aggregation network (PANet) as its neck. The transmission of low-level features is boosted by the use of a novel feature pyramid network (FPN) structure by PANet with an improved bottom-up path. In addition, adaptive feature pooling, which connects the feature grid and all feature levels, enables each feature level's vital information to spread instantly to the next subnetwork. In lower layers, PANet enhances precise localization signals, which can increase the object's location accuracy, as shown in figure 2.6. Thirdly, to provide multi-scale prediction, the YOLO layer, the Head of YOLOv5, generates feature maps in three distinct sizes allowing the model to handle tiny, medium, and large objects.

2.3.5 YOLOv7

The real-time object identification model with the highest speed and accuracy is YOLOv7. Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao published the official YOLOv7 paper titled "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors" in July 2022. [36]

2.3.5.1 YOLOv7 Architecture

- **Extended Efficient Layer Aggregation Network (E-ELAN):** The computational building piece of the YOLOv7 backbone is the E-ELAN. Figure 2.7 shows the E-ELAN architecture, which allows the framework to learn more effectively where it's constructed using the ELAN computational block. It draws influence from earlier studies on network effectiveness. It was created by looking at the following elements that affect speed and accuracy.
 - Memory access cost
 - I/O channel ratio
 - Element-wise operation
 - Activations
 - Gradient path
- **Model Scaling for Concatenation-based Models:** Object detection models are often published as a succession of models, scaling up and down in size, because different applications demand varying accuracy and inference speeds. Typically, object detection models incorporate the depth of the network, the

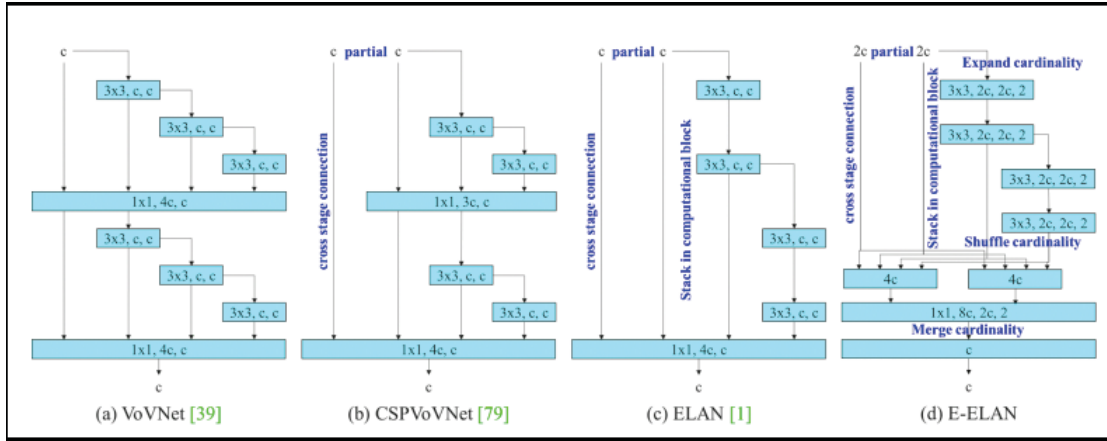


Figure 2.7: E-ELAN [25]

width of the network, and the resolution that the network is trained on. The YOLOv7 algorithm uses a compound scaling method, which does not disturb the initial design of the model and gives optimal results.

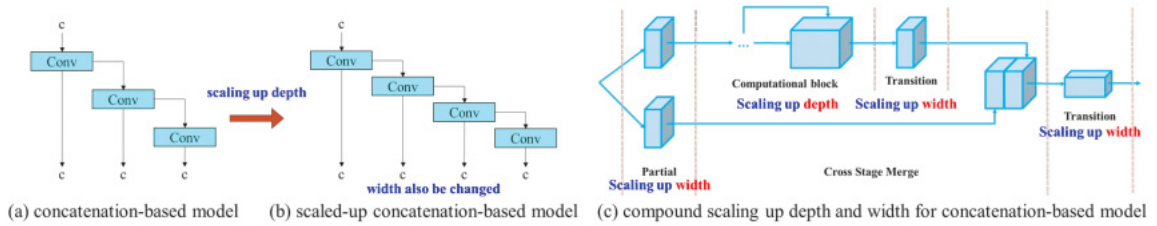


Figure 2.8: Compound Scaling [25]

Trainable Bag of Freebies: Features that boost accuracy without slowing down detection speed are called bag-of-freebies. Because of this, YOLOv7 is faster and more accurate than earlier YOLO versions. The YOLOv4 paper first used the concept. The methods of the trainable bag of freebies are,

- **Planned re-parameterized convolution:** Re-parameterization is a technique performed after training to improve the model. It lengthens training duration while enhancing inference results. Two types of re-parametrization are used to finish models: model level and module level ensemble.
 - **Model level re-parametrization:** This can be accomplished in two ways.
 - * Train several models with different training data but the same settings. The final model is created by averaging their weights.
 - * Take the weights of models at different epochs and average them.

- **Module level re-parameterization:** This method divides the procedure into segments, as shown in figure 2.9. The results are combined to form the final model.

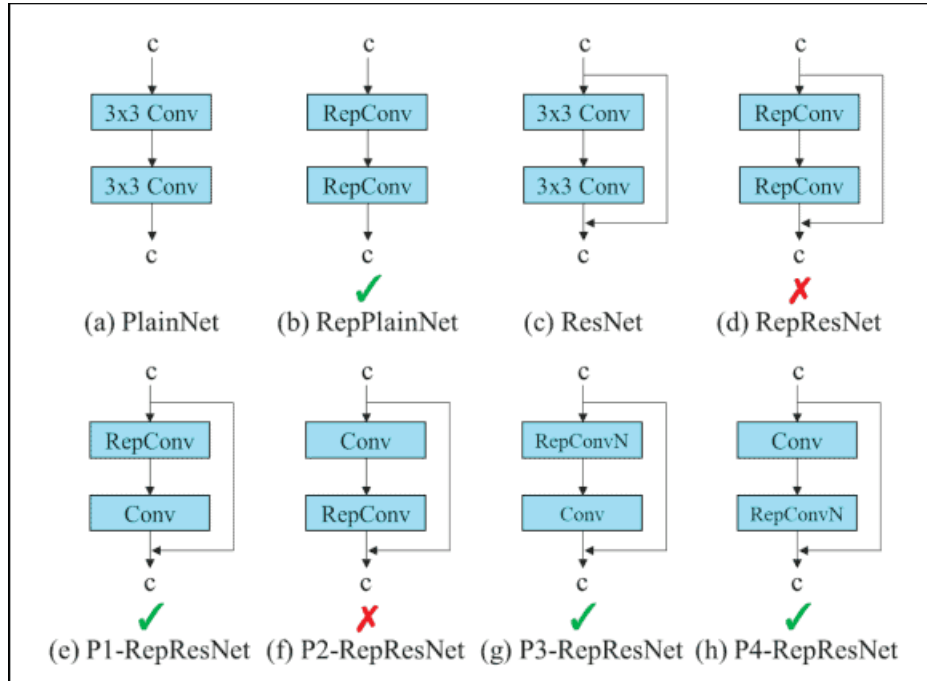


Figure 2.9: YOLOv7 Module planned re-parameterization [25]

- **Coarse for auxiliary and fine for lead loss:** During training, the coarse detection head is used as an auxiliary to the fine detection head, with the lead loss being the loss generated by the fine detection head. The lead loss is the primary objective in training, and the coarse auxiliary loss helps to guide the training process toward the desired lead loss, as shown in figure 2.10. This can improve the overall accuracy and stability of the model training.

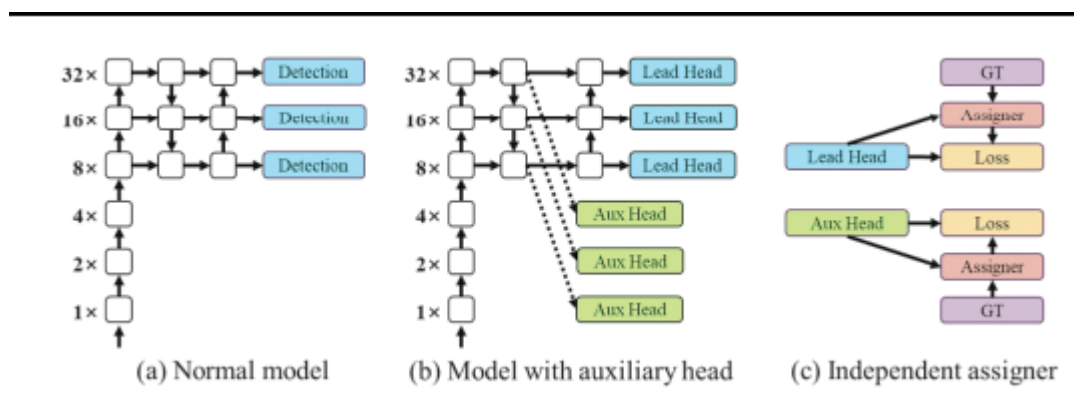


Figure 2.10: Coarse-to-fine auxiliary head in the YOLOv7 network [25]

Label Assigner is a technique that evaluates network prediction outcomes and ground truth before assigning soft labels, which is visually explained in figure 2.11. The label assigner generates soft and coarse labels rather than hard ones. The Lead Head Guided Label Assigner encapsulates the three ideas listed below.

- Lead Head
- Auxiliary Head
- Soft Label Assigner

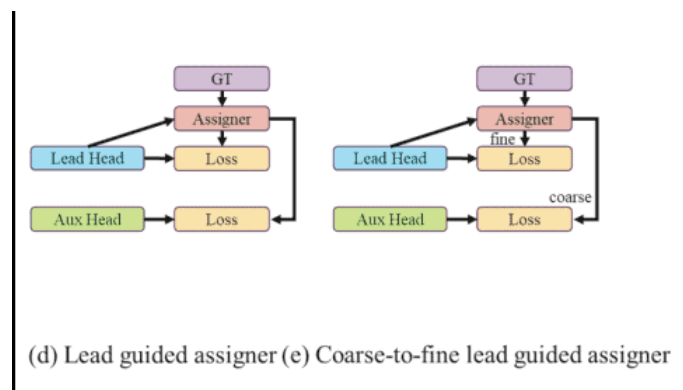


Figure 2.11: YOLOv7 Label Assigner [25]

The YOLOv7 network’s Lead Head forecasts the outcomes. These final results are used to generate soft labels. The critical point is that the loss is estimated for both the lead head and the auxiliary head using identical soft labels. Ultimately, the soft labels are used to instruct both heads.

There are two sets of soft labels created.

- A fine label for training the lead head.
- A collection of coarse labels for training the auxiliary Head.

2.4 The Handwritten Digit

The capacity of computers to recognize human handwritten digits is known as handwritten digit recognition. It is difficult to work for the computer since handwritten digits are erroneous and can be produced with various tastes. This challenge can be solved by using handwritten digit recognition, which takes a picture of a digit to identify the digit in the image. [37] [38] There are offline and online techniques to recognize a handwritten digit.

- **Offline Recognition:** Offline handwriting recognition automatically converts text in an image into letter codes that can be used in computer and text-processing applications. This form’s data is regarded as a static representation

of handwriting. Because different people have different handwriting styles, offline handwriting recognition is difficult. [39] [40] [41]

- **Traditional Technique:** Traditional handwritten recognition techniques, such as Hidden Markov Models (HMMs) and Dynamic Time Warping (DTW), use statistical and rule-based methods to recognize handwritten characters. They require manual feature extraction and depend on predefined class templates to match and recognize patterns.
- **Modern Technique:** Modern handwritten recognition techniques like Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) use deep learning algorithms to recognize handwriting. They can automatically extract relevant features from the input data and learn to recognize patterns in a more flexible and scalable way.
- **Online recognition:** Online handwriting recognition involves converting text as typed on a specific digitizer, where a sensor detects pen-tip movements and switches from pen-up to pen-down. This form of data is known as digital ink and can be thought of as a digital version of handwriting. Letter codes are created from the resulting signal in text processing and computer programs [42].

The handwritten digit detection from Swedish historical documents is an offline recognition process as the data is extracted from scripts and modern techniques are used in this paper as they can handle variations in writing style and scale more efficiently than traditional methods.

In [43] Hongjian Zhan et al., proposed a method for recognizing handwritten digit strings using a new CNN-based architecture. The network design has three levels from bottom to top: feature extraction layers, feature dimension transposition layers, and an output layer. They used DenseNet, and CTC approaches for the above sections of the network architecture. The authors have evaluated these architectures using ORAND-CAR Datasets, with recognition rates of 92.2% and 94.02%, respectively.

In [44], Anuran Chakraborty et al., proposed a method for recognizing irregular digits and touching digits in a string. They discovered that while the various models had great recognition accuracy on some standard datasets, they fell short in recognizing Handwritten digit strings. Their method begins with a deep autoencoder-based segmentation technique to isolate digits from a handwritten string, then passes on to a Residual Network-based recognition model. They assessed their results using the Handwritten Digit Strings (HDS) database using Computer Vision Lab (CVL). Their future effort will be to improve the segmentation approach, which now cannot distinguish some of the digits written in a single pass in a digit string.

Authors [45] proposed a method for digit string recognition using a hybrid principal component analysis network(PCANet) and support vector machine (SVM) classifier called PCA-SVMNet. This process has taken place without using any segmentation techniques. PCANet has only two convolutional layers, so the authors have combined the PCA-SVMNet model with a connected layer. They have trained this resultant architecture using the natural and synthetic attached digits. The cascade of the PCA-SVMNet classifier, which is used for recognizing the connected and isolated digits, has high efficiency for recognizing the connected digits and the unknown handwritten digit string.

In [46], authors have proposed a new architecture called Gated-CNN-BGRU to compare the recognition methods for automatic handwriting recognition systems with the help of data from an international conference. From the International Conference on Frontiers of Handwriting Recognition (ICFHR), they have taken metrics, datasets, and recognition methods for comparison and a private dataset, 11 different approaches from the state-of-the-art HDSR, and two optical models from the state-of-the-art in HTR. The result shows that the optical model has been efficient in recognizing the digits in the images even if there is high noise. Their model showed

a precision of 93.54%.

The study [47] suggests a novel method for improving handwriting images that uses Gaussian Mixture Model. Two quantitative techniques, discrete entropy (DE) and edge-based contrast measure (EBCM), are used to quantify the quality of each patch as a fixed-size window passes over the handwritten image. The outcomes are used in the unsupervised learning approach by assigning the handwriting quality using k-means clustering. To remove the artifacts that represent the final image, GMM is utilized as a final step. For various datasets, the suggested method has been contrasted with the other methods. According to the results, the proposed method outperforms the currently used contrast enhancement techniques in handwriting.

Authors [48] also proposed DIGI-Net, which is a deep convolutional network. Here they experimented with handwritten, natural images and printed font from the three datasets MNIST, CVL single digit dataset, and digits of Chars74K dataset. The CNN architecture used has a great learning ability. The features from printed font digit images improve the accuracy of handwritten digits and raw image digits. It has an accuracy of 99.11%, 93.29%, and 97.60% for the respective datasets.

To get an efficient and improved version of handwritten digit recognition, the authors [49] proposed a method having a convolutional neural network. They opted for the MNIST dataset for training and testing the proposed model and the DL4J framework for handwritten digit recognition. The system reduced the computational time for training and testing the data. The accuracy for the system shows highest as up to 99.21%, which was verified repeatedly by checking the system by changing the CNN layers.

Siham Tabik et al. [50] proposed image preprocessing for convolutional neural networks. They analyzed the MNIST dataset using different preprocessing techniques on the performance of three CNNs, LeNet, Network3, and DropConnect, together with their ensembles. The analyzed transformations are centering, elastic deformation, translation, rotation, and different combinations of them. Based on their findings, they believe that data-preprocessing approaches like the combination of elastic deformation and rotation, as well as ensembles, have a huge amount of potential.

The authors [51] suggest a preprocessing stage to improve the bank of features which can extract from binary images and aid in boosting the precision of pattern recognition algorithms using shape growth pattern analysis. The authors have implemented the categorization procedure using CNN and Random Forest. The results demonstrate a significant improvement in classification accuracy of 90.6 percentage compared to previous methods.

The study [52] compares the performance of three machine learning methods: Support Vector Machines (SVM), Artificial Neural Networks (ANN), and Convolutional Neural Networks (CNN) to determine the best approach for recognizing handwritten digits in document images. The methods use HOG feature extraction and deep learning techniques to process the digit images. The results show that the CNN

method outperforms the SVM and ANN methods, with a recognition rate of 71

The author [53] focuses on building a segmentation-free method for handwritten multi-digit string recognition in the context of document digitization. The study aims to evaluate and compare three models: VGG-16, CRNN, and 4C, and to research the effect of different training sets on the model performance by using the accuracy, ANLD, and confusion matrix as evaluation metrics. The results show that the 4C framework based on the improved VGG network performs the best in all aspects. The results show that the 4C framework built on the enhanced VGG has the best outcomes.

For detecting and recognizing handwritten digits in historical documents, authors [54] introduce a deep learning architecture called DIGINET and a dataset called DIDA to train the model. Data in the DIDA dataset is collected from the historical Swedish handwritten document images. DIGINET consists of DIGITNET-dect and DIGITNET-rec, which are used to extract features for detecting the digits and recognizing digits using CNN architecture, respectively. The complete experiment shows that the DIGINET has the highest detection with 76.84% accuracy and recognition with 97.12% accuracy compared to other models.

3.1 Research Gap

In previous studies, authors [45] [49] have mainly focused on digit detection and recognition of modern handwritten styles. They have worked on denoised, cleaned, size normalized datasets. The present digit detection and recognition methods are limited to modern handwritten digits with no artifacts. Historical documents are essential as they are assets for future generations and must be kept safe from damage and loss. Automating the process of identifying the data from the records and storing them is a significant step to make. The subsequent research gaps from previous studies are listed below:

- In [43] [45] [47] [48] [49], the authors have implemented deep learning algorithms for digit detection on cleaned, denoised datasets with modern handwriting styles. These approaches work well with existing digit datasets but not with historical documents. With the help of the DIDA dataset, we implement advanced deep learning algorithms to detect and recognize multi-digit strings and compare them based on metric evaluation measures. The authors of the paper [52] [53] have worked on recognising the handwritten digits but used ML algorithms or basic DL algorithms in their research. The digits in historical documents from DIDA dataset [10] have multiple issues such as single-touch, multi-touch, broken, Degradation, and Fainted. Apart from that, the digits were written in ancient handwriting styles.
- There is a shortage of experimentation based on advanced deep learning methods to detect digit strings on historical documents from the DIDA dataset [10].
- The authors compared advanced deep learning algorithms [49] on MNIST

dataset [7]. In our research, we will implement advanced deep learning algorithms for detecting and recognizing multi-digit strings on historical documents collected by DIDA dataset [10].

This section has a detailed implementation of chosen research methods; datasets used for training and testing our models, and an evaluation strategy to analyze the models trained and tested with the DIDA dataset. The most used empirical methods are Experiment [55], Case study [56], and Survey. For our research, we have chosen a literature review and Experimental study as we use different advanced deep learning algorithms by training and testing the data from the DIDA dataset. The Literature review helps us to gain knowledge about different advanced deep learning algorithms for training and testing the DIDA dataset [10]. Our quantitative dataset (augmented and digit string images) is why we chose experimental study as our research method. The reason for not choosing the following methods:

A **Case Study** is not suitable for this research as it is related to interviews and archival data, which are not needed to know the efficiency of an algorithm for detecting digits.

The **Survey** includes a set of questions and deals with opinionated views. The potential for in-depth exploration of influential factors in the study is limited, and Survey collects data. Still, it cannot give a hypothetical result as an experiment.

4.1 Literature Review

The literature review is to gain knowledge of the existing research relevant to a specific study area. We have conducted this method to find the practical deep-learning algorithms that solve our RQ1 for detecting and recognizing handwritten digit strings. We have learned about different deep-learning algorithms for object detection and data preprocessing techniques.

Our primary step while conducting a literature review is forming a set of keywords relevant to our research area. We use these keywords in our research to find the articles that help us find deep learning algorithms for digit string detection from the databases such as Springer, IEEE Xplore, Scopus, Google Scholar, etc. The search string used for finding articles for our study is presented below :

("Deep learning algorithms" OR "Deep Learning models", "Handwritten digit detection" OR "Handwritten digit recognition" OR "Comparison of advanced deep learning algorithms" OR "Object detection algorithms" OR "Machine Learning" OR "Neural Networks")

We employed inclusion and exclusion criteria, as well as the snowball sampling approach [55] to focus our results on the most relevant articles. The snowball method starts with a small initial sample and builds upon it by recruiting more participants.

Inclusion Criteria:

- Articles that are published in the English language.
- Articles which are available in Full text.
- Articles that include either "Deep Learning" or "Object Detection".
- Articles must have been published in the last 20 years.

Exclusion Criteria:

- Articles that deal with subjects besides machine learning.
- Articles that lack full-text access.

4.2 Experimentation

Experimentation is a highly efficient research method to know the cause-and-effect relationships between variables and provides a systematic approach to measuring the outcome. [55]. The experimental study is suitable for our research to resolve RQ2 and RQ3, where we train different advanced deep learning models to detect handwritten multi-digit strings with the DIDA dataset III and then measure the efficiency of each model using Evaluation metrics.

Historical handwritten digit images are more complicated to process when compared to other digit images as handwritten digits have many issues which make them difficult to identify, as mentioned earlier. The algorithms should overcome these issues and accurately detect the numbers in the multi-digit strings as the experiment aims to detect digits in a multi-digit string. The problem becomes more complicated as the digits in the strings may collide with each other while being written, which need to be visible correctly, are darkened, and so on.

We propose a solution with the experimental study to address the problems of detecting a digit string. To accomplish this, we train the model using the DIDA Dataset-III, a collection of augmented digit images representing all digit classes. We then evaluate the model using dataset-II, which contains multi-digit strings from the same dataset. Our conclusion is based on the evaluation of various advanced deep learning algorithms to determine which one has the highest efficiency in detecting and recognizing digit strings. The efficiency of each algorithm is measured using key metrics such as Precision, Recall, Accuracy, and F1-score.

We chose the particular Deep Learning algorithms based on the literature review done on previous studies. This section elaborates on the training and testing of algorithms which are discussed in the background section.

In our experimental study, the initial stage involves determining the dependent and independent variables. The independent variable, which is the subject of manipulation and alteration, has an impact on the dependent variable. The dependent variable is the variable that is being measured and evaluated.

Dependent variables : Training deep learning models, Evaluation metrics.

Independent variables: Standard Advanced Deep learning algorithms.

4.3 Tools and Libraries

In this research, we have picked up some deep learning algorithms for multi-digit string detection. We have implemented the algorithms using the python language and carried out our experiment in a Google Colab environment.

Hardware specification

CPU	Intel (R) Xeon(R) @ 2.30GHz with 2 cores
GPU	Nvidia T4
RAM	12GB
GPU memory	16GB

Table 4.1: Google Colab Pro Environment

Although our selected deep learning algorithms require the base libraries but also have different requirements based on their design architecture. Here we mention the core libraries and algorithm-specific dependencies.

4.3.1 Core Libraries

Pandas: Pandas library stores various analysis tools and high-level data structures. It provides data cleaning and processing and supports data conversion, visualization, and more. [58]

Numpy: Numpy is a most used python library that supports large matrices and multi-dimensional data. It provides mathematical functions for quick calculations. [60]

OpenCV: It is a computer vision library that gives computer vision applications a common framework and makes it easier to incorporate machine learning into commercial products. [57]

Matplotlib: Plotting numerical data is handled by this library. And for this reason, it is employed in data analysis. Additionally, an open-source library, it plots highly detailed images such as graphs, scatterplots, graphs, and pie charts. [61]

Sklearn: Sklearn is a popular Python Library for handling complex data. It is an open-source library that facilitates machine learning. It supports many supervised and unsupervised methods, including linear regression, classification, and clustering. This library is combined with both SciPy and Numpy. [59]

Tensorboard:As part of the machine learning workflow, TensorBoard is a tool for providing the measurements and visualizations required. It enables a variety of tasks, including visualizing the model graph, projecting embeddings into a lower dimensional space, and tracking experiment metrics like loss and accuracy. [62]

4.3.2 Algorithm specific libraries

Tensorflow:It is an open-source library utilized for complex calculations. Deep learning and machine learning algorithms both make use of it. There are numerous tensor operations in it. [63]

PyTorch:PyTorch is the most popular machine-learning library for tensor computation optimization. It helps with the resolution of issues related to neural network applications by having strong GPU acceleration and a rich API for tensor computations. [64]

Torchvision:Torchvision is a computer vision library that works together with PyTorch. There are tools for quick image and video transformations, pre-trained models that are widely used, and datasets. [64]

Keras:Google created the high-level Keras deep learning API to implement neural networks. It is used to simplify the implementation of neural networks and is written in Python. Additionally, multiple backend neural network computation is supported. [65]

4.4 Dataset

DIDA has the largest image-based historical handwritten digit dataset with 250k single digits and 200k multi digits, which are taken from the Swedish historical handwritten document images between 1800 and 1940 [10]. The digit images are recorded without size normalization and denoising, and the DIDA dataset is generated to resolve handwritten digit detection and recognition in the historical document images. DIDA dataset consists of three sub-datasets, and we use Datasets II and III for our Experimentation.

- **Dataset II :** This dataset consists of multi-digit samples with two variations. One is a fixed length, and the other is a variable-length string. This dataset has six numeral string issues: single-touch, multi-touch, broken, degradation, faded or invisibility, and no artifacts. In this dataset, the digit images are used as test images for each trained detection model for detecting the digits in the multi-digit string.

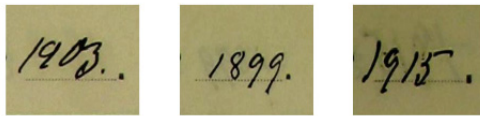


Figure 4.1: Fixed length multi-digit string images [10]

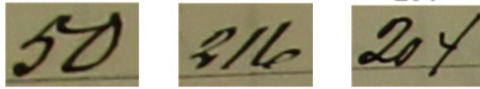
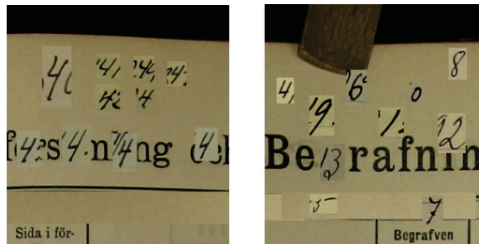


Figure 4.2: variable length multi-digit string images [10]

- **Dataset III** : All digit classes' augmented digit images can be found in this dataset. These images are formed by superimposing every ten digits from Dataset I on cropped images from historical handwritten documents where initial digits are not included. The authors generate this dataset [54] to train their Digitnet Network to recognize digit strings. We have used this dataset to train our Advanced deep-learning models for detecting and recognizing historical handwritten digit strings.

Figure 4.3: single handwritten digit samples in the 416×416 images [10]

4.5 Research Design

This study mainly works with the detection of handwritten digits in images. The document images in the DIDA dataset are used for training and testing the advanced deep-learning models. We use the following research design for implementing our advanced deep learning algorithms. Ultimately, we evaluate the models based on the metrics such as Accuracy, Precision, and Recall and compare them to determine the best deep learning model among the other models for detecting and recognizing historical handwritten document images from DIDA dataset [10].

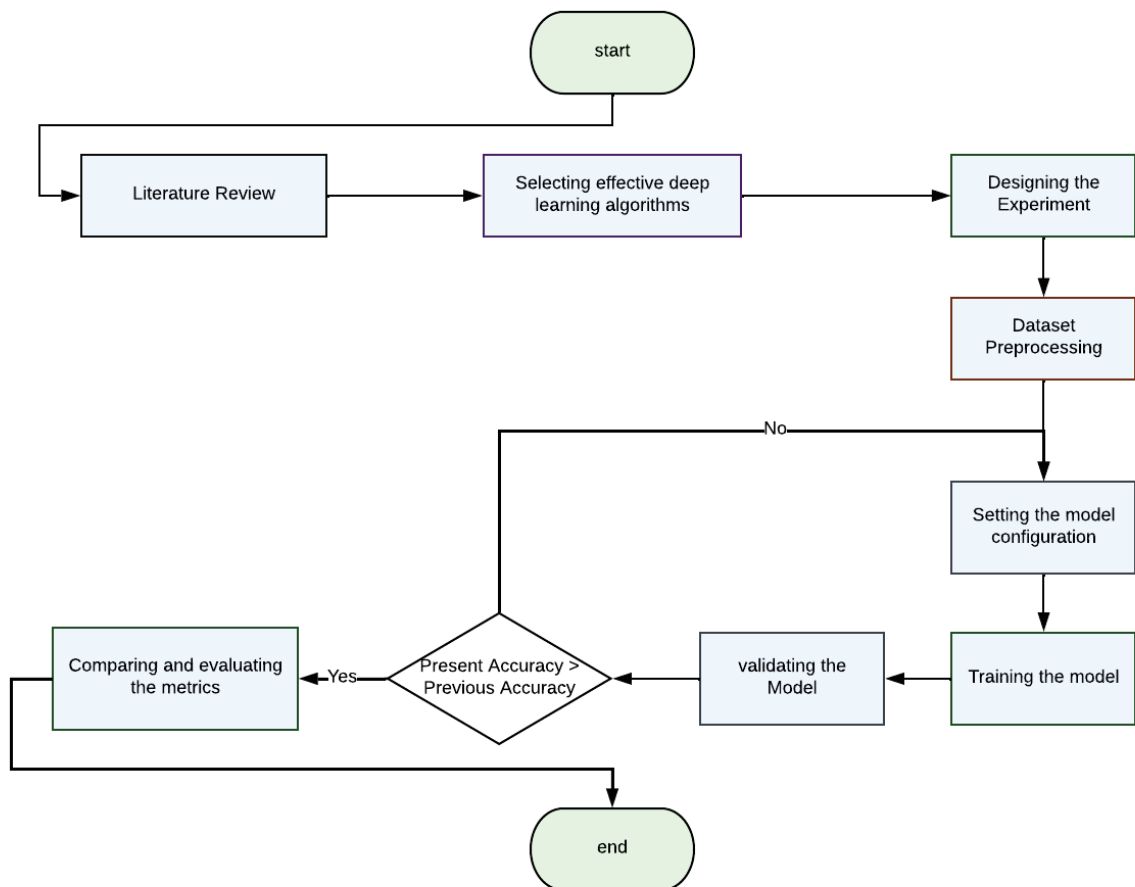


Figure 4.4: Research Design

The research process is done step by step as follows,

- The research starts with a literature review to find suitable deep-learning algorithms.
- The next step is to design the experiment with the required data and tools.
- Preprocessing the data in the dataset by properly modifying the data as per requirements.

- Implementing each model with the preprocessed data by setting some model configurations to detect the digits in the images.
- Training every model using 13,000 images, 4,000 images for testing, and 2,000 images for validating the model.
- If the validated results are better than the previous one then move to the final results otherwise change the model configuration settings and continue to the next steps.
- Finally, compare all the algorithms with one another with their results to know the better among all four algorithms.

4.6 Implementation

The implementation of a deep learning model involves several steps, including data preprocessing, model configuration, model training and validation, and model evaluation. In the data preparation phase, the dataset of images and annotations must be collected and preprocessed to ensure that it is suitable for training the model. During the model training phase, the algorithm is fed by the preprocessed data, and the parameters of the model are optimized to avoid the overfitting of the deep learning model and obtain the optimum accuracy of the model.

4.6.1 Data Preprocessing

The first step is **Data Preprocessing**: Data preprocessing is crucial, where the raw data is manipulated according to the output's requirements, improving the model's performance. We implemented Image preprocessing using these operations: Labeling, Image Resizing, and Data Augmentation.

Labeling- Labeling in preprocessing refers to the process of assigning a class or category label to each instance in a dataset. Labeling can be performed manually by using tools. Roboflow application [68] is used for labeling the digits, where the different algorithm has different file format. These file formats are provided in "Appendix A". Figure A.1 shows the YOLOv5 and YOLOv7 file, which is in "text" format. Figure A.2 shows the RetinaNet file in "CSV" format. Faster RCNN is shown in ".tfrecord" file format in figure A.3.

Image resizing- This step resizes the filtered clean Image using Normalization, also called re-scaling. Different size of the Image indicates a difference in pixels in the Image; depending on the pixels learning rate will differ.

Data Augmentation- It increases the model's performance as the process prevents the network from learning the unnecessary features in the Image. The techniques used in this process are vertical and horizontal flipping, cropping, and rotation.

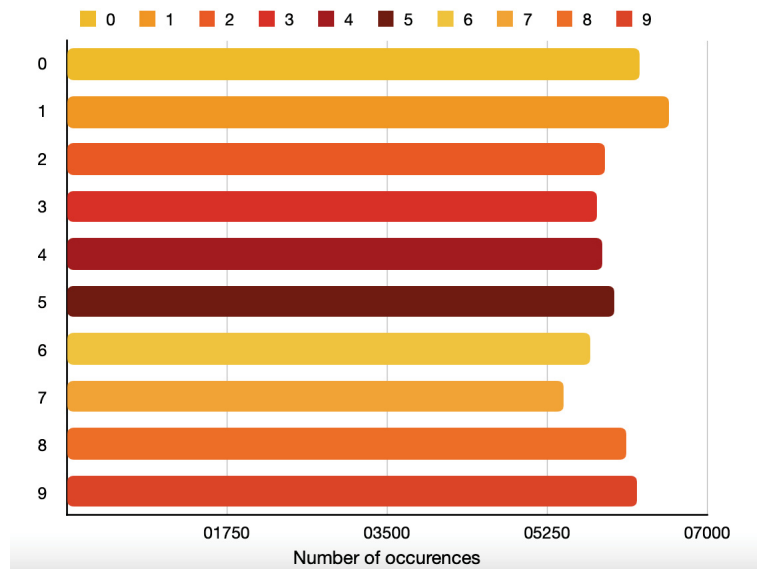


Figure 4.5: Digits frequency in the training data

4.6.2 Model Training

A deep learning algorithm is fed training data so it can learn during the process of "training" a deep learning model. All four algorithms are trained using 13,000 images and then tested using 4,000 images to evaluate the output. Our study uses YOLOv5, Faster R-CNN, YOLOv7, and Retinanet deep learning algorithms to train on the DIDA dataset III. These algorithms are trained by using images with an image resolution of 416x416. Every deep learning model, in essence, every machine learning model, goes through a set of steps typically the same when training a model. During model training, the following steps are taken:

- **Setting up the Environment** : Our priority is to set up the Environment, which involves importing the necessary libraries and dependencies for training the model.
- **Model Configuration** : Every deep learning algorithm has a configuration file where we can change the deep learning model's hyperparameters. These are variables that have an impact on the training process and ultimately determine the performance of the model. The reason behind choosing these hyperparameters because the authors [66] mentioned that the batch size and learning rate would be considered as a critical options for obtaining an optimized deep learning model. The following hyperparameters used in our study are.
 - **Epochs** : Epoch refers to a single iteration over the entire training dataset in a deep learning model.
 - **Batch Size** : Batch size defines the number of digit images processed at a time in a single forward/backward pass.
 - **Learning Rate**: Learning rate is a hyperparameter that determines the step size at which the optimizer makes updates to the model parameters during training.

The selection of these hyperparameters is a crucial step in the training process and is performed by searching through a defined space of possibilities. The final values chosen for these hyperparameters are based on the results of the training and validation processes, and can be adjusted as necessary to optimize the model's performance.

Hyperparameter	Attribute	Search Space	Selected value
Epochs	Fixed	[100]	100
Batch Size	Variable	[8, 16]	8
Learning Rate	Variable	[0.002, 0.003, 0.0035]	0.002

Table 4.2: Details of the hyperparameter search space are displayed in the table.

The range of all potential solutions that must be searched is defined by the search space. The search space must be planned so that it is neither too tiny to exclude many satisfying models nor too large to require more effort and resources. We have picked the Learning rate search space they were near to default values of the chosen deep learning models. We have picked the default epoch variable (100). We have used google colab as our experimental environment the batch size is 8 and 16 were appropriate choices since anymore would be computationally taxing on the environment.

- **Implementation of tensorboard :** To analyze the model training, we receive live metric graphs, which is crucial. This makes it easier to recognize the model's behavior, and we then change the values of the hyperparameters.
- **confidence score threshold :** Confidence score in object detection refers to the probability that a detection is accurate. Higher confidence scores indicate a higher degree of certainty that the detection is correct, while lower scores indicate more uncertainty. Confidence scores can be used to filter out low confidence detections and improve overall accuracy [66] For our Experiment we have taken the default confidence score threshold (0.5) as it provides a good balance between reducing false detections and ensuring that the majority of true detections are kept. A threshold of 0.5 means that only detections with a probability score higher than 0.5 will be considered as valid detections.
- **Saving the model:** Later, we use a package associated with the created model. For saving the deep learning model, each library has a separate sub-package. We reload this saved model into the notebook and test it using the handwritten multi-digit images.

4.6.3 Model Evaluation

Digit string detection and recognition through different advanced deep learning algorithms give different results. The detection rate may vary for each algorithm. The performance of the detection algorithms to detect the digit string can be measured

using some factors. The metrics for evaluating the algorithm's efficiency are Precision, Recall, Accuracy and F1-score.

Precision - It is used to measure the correct predictions by calculating the division of true positives by all the positive predictions (true positive plus false positive). In comparison with our study, this represents the proportion of correctly detected digits among all the digit predictions of a specific digit [67].

$$Precision = \frac{TP}{TP + FP} \quad (4.1)$$

Recall - Recall is a metric used in evaluation of binary classification models and measures the proportion of actual positive instances that are correctly detected by the model. In our study, this represents the proportion of correctly detected digits among all the actual instances of a digit [67].

$$Recall = \frac{TP}{TP + FN} \quad (4.2)$$

Accuracy - Accuracy represents the proportion of correctly detected digits among all the actual instances of a digit [67].

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \quad (4.3)$$

F1score - F1 score is a balanced measure of precision and recall, computed as the harmonic mean of the two, used to evaluate the performance of a deep learning model. [67]

$$F_1score = \frac{TP}{TP + \frac{1}{2}(FP + FN)} \quad (4.4)$$

In this chapter, we present the findings from the Literature Review, the Experiments, and the analysis done to address the research questions.

5.1 Literature Review

In this section, we present the results of the research papers and articles within the area of our current research. The findings from the literature review provide context and background information about the topic being studied.

Article	Results
An improved faster-RCNN model for handwritten character recognition. [69]	The authors of the paper introduced a model for handwritten character recognition. The model implementation is to obtain the region of interest by creating annotations. Then the in-depth features are then computed using an upgraded Faster-RCNN that incorporates DenseNet-41. The digits are then localized and divided into ten classes using the regressor and classification layers. The results show that the accuracy obtained for handwritten character recognition using Faster R-CNN is 99.78%.
Automatic Container Code Recognition via Faster-RCNN. [70]	This study uses Faster-RCNN to automatically find and identify container codes. Candidate characters are passed to a post-processing program that uses binary search as its foundation. Faster-RCNN then uses these characters to acquire the container code in an effective and efficient manner. Experimental findings support the approach suggested in the research, which achieves an overall accuracy of 97.71% on a dataset containing 831 container codes.

Article	Results
Faster R-CNN: an Approach to Real-Time Object Detection. [71]	The authors of this study justify the Faster R-CNN used for the detection of the traffic indicators like traffic lights and STOP signs. The algorithm is taken as it is a suitable algorithm for the application in terms of detection accuracy and speed. Experimental results show that 98% mean average precision is achieved using the Faster R-CNN algorithm.
Automatic License Plate Recognition via sliding-window darknet-YOLO deep learning. [72]	The study is to detect the license plate of the car automatically. The method used to detect is a sliding window, where it takes the digits in the license one by one, and then each window is taken by the YOLO framework for detection. The paper says that the system achieved approximately 98.22% accuracy on license plate detection using the YOLO algorithm.
Improved Ship Detection Algorithm from Satellite Images Using YOLOv7 and Graph Neural Network. [73]	The authors of this paper have provided a model that uses YOLOV7 and GNN to identify ships in satellite photos. The learning rate, batch sizes, and optimization choices are the three hyperparameters used to create this system. According to the testing results, the proposed model outperforms the YOLOv7 algorithm's predecessor with a success rate of 93.4%.
A comprehensive comparison of end-to-end approaches for handwritten digit string recognition. [74]	This study is to conclude with the best algorithm to solve the HDSR (Handwritten Digit String Recognition) problems. The study compares YOLO, RetinaNet, and CRNN algorithms, considering the challenging datasets taken for training and testing. The authors claim that the YOLO model has achieved an accuracy of 97% for handwritten digit string recognition, which is the highest compared to other algorithms.
Handwritten Text and Digit Classification on Rwandan Perioperative Flowsheets via YOLOv5. [75]	The paper shows the detection of handwritten documents to digitalize them. The implementation has been done on the 363 perioperative flowsheets of the drug and physiological indicator portions from the University Teaching Hospital of Kigali in Rwanda. The authors stated that the detection of handwritten documents using YOLOV5 had shown outstanding results. For the detection of word objects by the model, the class labels of the YOLOv5 model have been combined into a single class.

Article	Results
Object Detection with RetinaNet on Aerial Imagery: The Algarve Landscape. [76]	The paper presents a model to detect the swimming pools in satellite images using the RetinaNet algorithm. The algorithm has been used with three different backbone architectures and compared to see which handles the class imbalances and which is good at the detection of smaller objects. ResNet152 backbone architecture of RetinaNet has shown high performance in detecting the swimming pools in the images.
Deep RetinaNet-Based Detection and Classification of Road Markings by Visible Light Camera Sensors. [77]	In this paper, the authors have proposed a method to detect road markings by visible light camera sensors using RetinaNet. Three datasets - Malaga urban dataset, the Cambridge dataset, and the Daimler dataset have been used to test the efficiency. It has been proved that the RetinaNet algorithm has been better in accuracy and speed than other algorithms.

Table 5.1: The results of the Literature Review are shown in the table.

5.2 Literature Review Analysis

From the literature review table 5.1 we have observed that the performance of every algorithm has been compared using metrics. With that in mind we selected the set of suitable algorithms from each paper for our comparative study.

In the papers [76] [77], RetinaNet has been used for the object detection of satellite images and road markings. The authors stated it was worth considering the algorithm as it showed better results in the detection process, and the architecture of the algorithm is flexible. In paper [73], the detection of objects is from satellite images. In the paper, YOLOv7 achieved an accuracy of 93.4% with satellite images. The papers [69] [70] [72] [74] [75] demonstrated successful digit and text recognition with different datasets; in these papers, Faster R-CNN, YOLOv5, YOLOv7, and RetinaNet have been proven to have high accuracy in digit and text detection. With the overall literature review and analysis, the following advanced deep learning algorithms were selected for our comparative study

- Faster R-CNN
- YOLOv5
- YOLOv7
- RetinaNet

5.3 Experimental Results

After all the implementations and training of the four algorithms with different architectures, the results have been obtained on recognizing and detecting the handwritten digits. The efficiency of the model is evaluated using a confusion matrix by visualization. Performance metrics like recall, precision, and accuracy are derived from the confusion matrix with true positives, true negatives, false positives, and false negatives. The results of the metrics comparison justify which algorithm is better in performance in detecting handwritten digits.

5.3.1 Results of Faster R-CNN

The results obtained show that the Faster R-CNN was able to recognize and detect the handwritten digits from the historical document images, as shown in figure 5.1. The algorithm was able to process and implement all types of digits with different challenges. The confusion matrix representation is shown in figure 5.2. We have achieved these test results by training the model with hyperparameters learning rate and batch size being 0.002 and 16.

5.3.2 Results of RetinaNet

The results in figure 5.3 show that the RetinaNet is successful in recognizing and detecting handwritten digits in historical document images. Each problematic digit in the image is preprocessed and used by the model. Figure 5.4 displays the confusion matrix results for the RetinaNet algorithm. We have achieved these test results by training the model with hyperparameters learning rate and batch size being 0.003 and 16.

5.3.3 Results of YOLOv5

As can be seen in figure 5.5 from the results, the YOLOv5 was successful in identifying and detecting handwritten digits in historical document images. All digits with various problems are preprocessed and used by the model. The confusion matrix results for the YOLOv5 algorithm are given in figure 5.6. We have achieved these test results by training the model with hyperparameters learning rate and batch size being 0.003 and 16.

5.3.4 Results of YOLOv7

The YOLOv7 proved successful in identifying and detecting handwritten numbers in historical document images, as seen in the results in figure 5.7. The model pre-processes and uses every digit in the image, which has issues detecting. Figure 5.8 includes the YOLOv7 algorithm's confusion matrix findings. We have achieved these test results by training the model with hyperparameters learning rate and batch size being 0.0035 and 16.

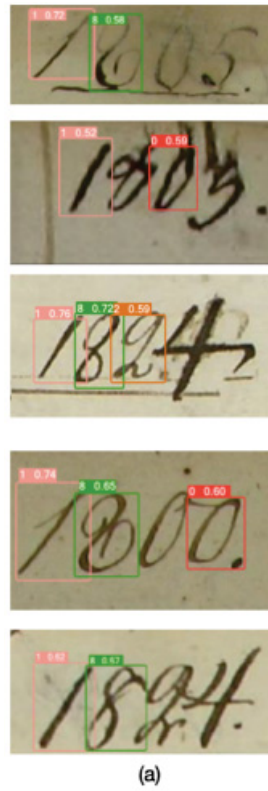


Figure 5.1: Faster R-CNN detection results

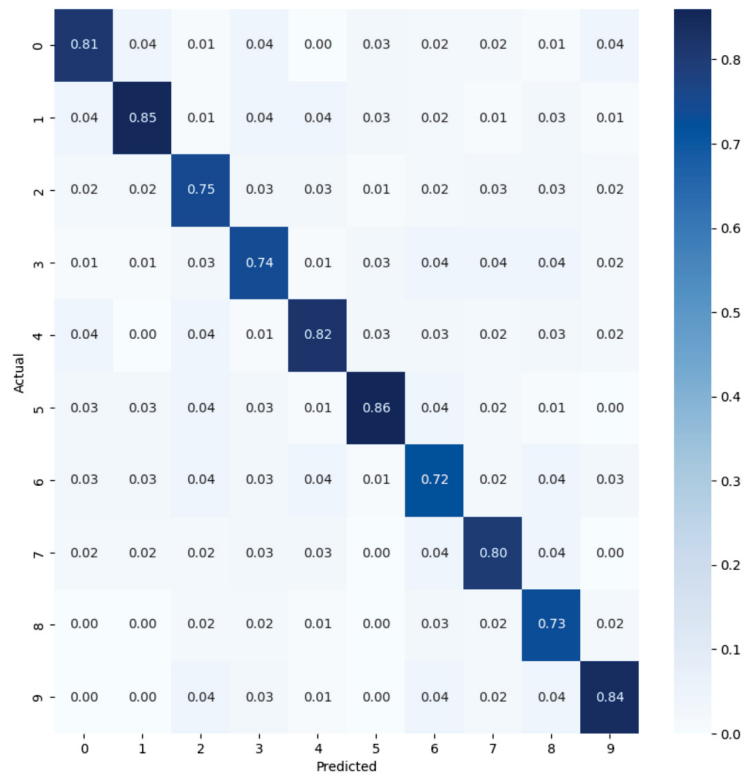


Figure 5.2: Faster-RCNN's Confusion Matrix

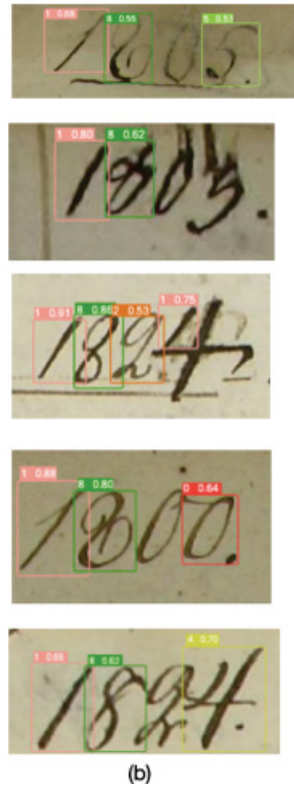


Figure 5.3: RetinaNet detection results

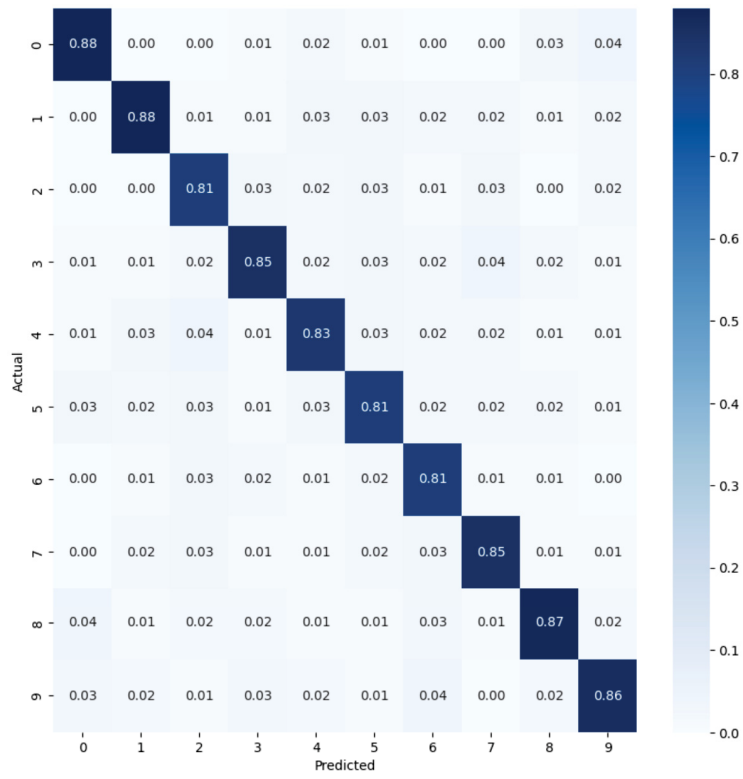


Figure 5.4: RetinaNet’s Confusion Matrix

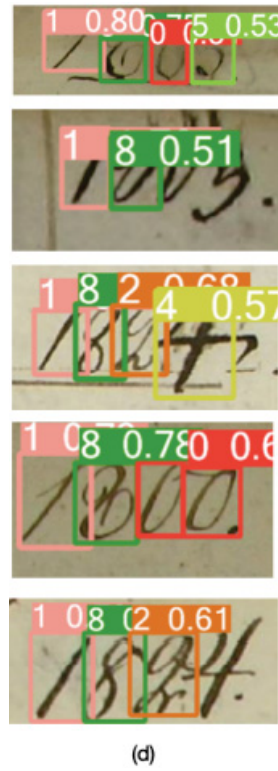


Figure 5.5: YOLOv5 detection results

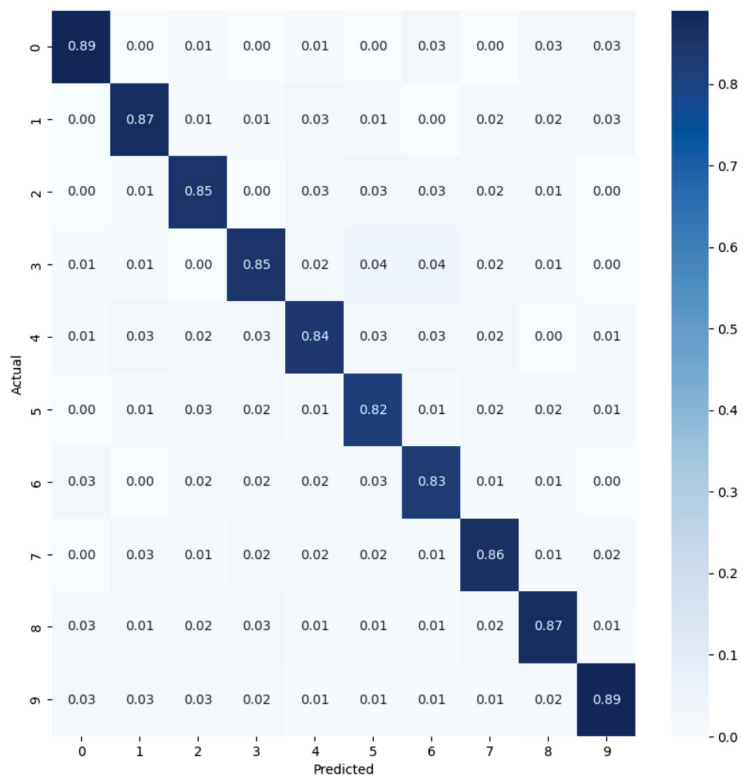


Figure 5.6: YOLOv5's Confusion Matrix

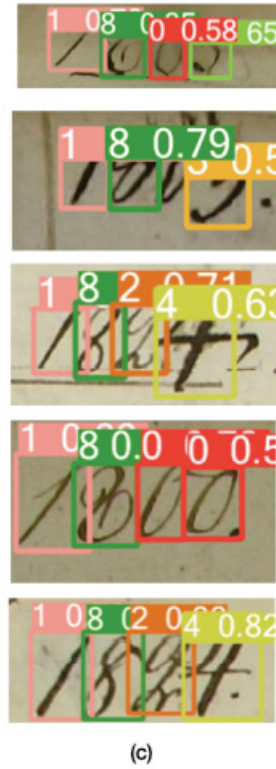


Figure 5.7: YOLOv7 detection results



Figure 5.8: YOLOv7's Confusion Matrix

5.4 Analysis of Experimental Results

Algorithms	Accuracy	Precision	Recall	F1 score
Faster R-CNN	79.21%	70.98%	77.3%	74.01%
RetinaNet	84.52%	75.3%	77.5%	76.4%
YOLOv5	85.7%	93.1%	81.3%	86.8%
YOLOv7	88.7%	93.7%	83.9%	88.5%

Table 5.2: The results of digit detection are shown in the table.

5.4.1 Accuracy

Each algorithm's accuracy is listed in table 5.2. Accuracy is calculated by including the values of the total number of true positives, false positives, false negatives, and true negatives obtained by Faster R-CNN, YOLOv5, RetinaNet, and YOLOv7. This table demonstrates the accuracy of the Faster R-CNN, YOLOv5, RetinaNet, and YOLOv7 algorithms. The comparison between the accuracy of the algorithms shows that Faster R-CNN has lower accuracy and YOLOv7 has higher accuracy in detecting handwritten digits. As a result, it is abundantly obvious from the findings that YOLOv7 exceeds all other algorithms in terms of accuracy. Comparison results of accuracy obtained by all four algorithms are shown in figure 5.9 as a bar graph representation.

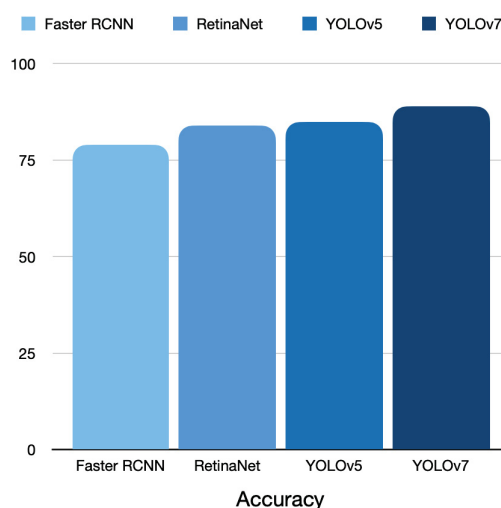


Figure 5.9: Comparison of detection accuracy for digit string images.

5.4.2 Precision

When the total number of true positives and false positives from the Faster R-CNN, YOLOv5, RetinaNet, and YOLOv7 algorithms are combined, the results of the precision are obtained, which are shown in table 5.2. It demonstrates that YOLOv7 has

higher precision than other algorithms. The visual representation of the comparison of the precision results is shown in figure 5.10.

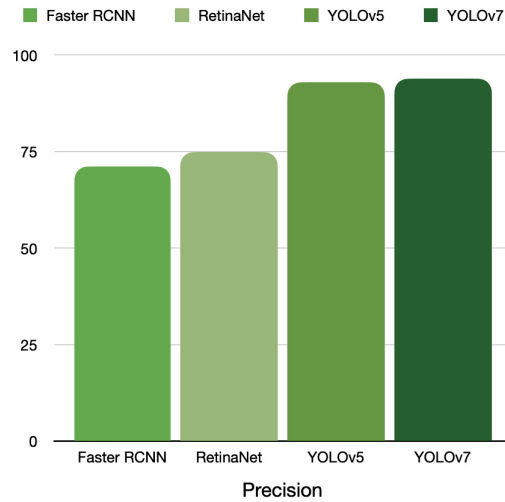


Figure 5.10: Comparison of detection precision for digit string images.

5.4.3 Recall

The total number of true positives and false negatives obtained from Faster R-CNN, YOLOv5, RetinaNet, and YOLOv7 algorithms are used to determine recall for respective algorithms. The recall results are listed in table 5.2. From this table, it is seen that the YOLOv7 algorithm has more recall percentage when compared to others. The comparison of the recall results is shown in figure 5.11 in the form of a graphical representation.

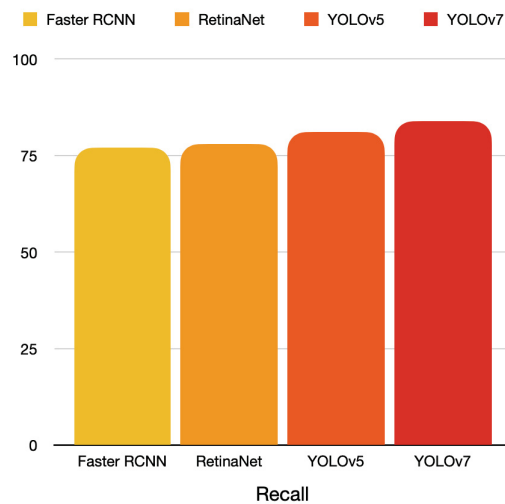


Figure 5.11: Comparison of detection Recall for digit string images.

5.4.4 F1 score

The total number of true positives, false positives and false negatives obtained from Faster R-CNN, YOLOv5, RetinaNet, and YOLOv7 algorithms are used to determine F1 score for respective algorithms. The F1 score results are listed in table 5.2. From this table, it is seen that the YOLOv7 algorithm has more recall percentage when compared to others. The comparison of the F1 score results is shown in figure 5.12 in the form of a graphical representation.

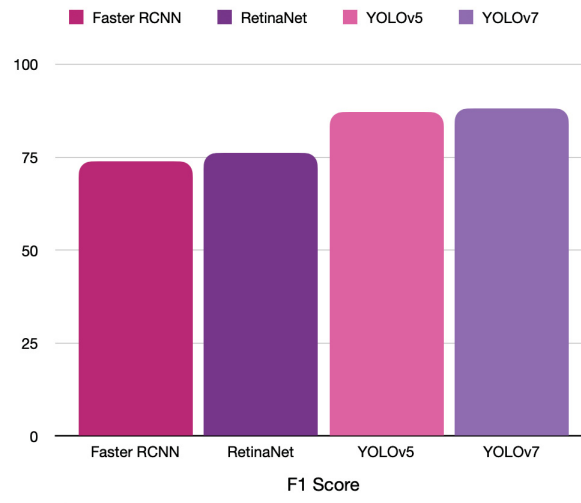


Figure 5.12: Comparison of detection F1 score for digit string images.

In this chapter, the aim is to present and explain the results of the research conducted. This includes a review of relevant literature, details about the experiments, and an analysis of the results collected to answer the research questions. The chapter will also discuss any potential risks to the research's validity or reliability, such as biases or the method's limitations. Discussing these threats is to provide a transparent and thorough examination of the study and identify areas where further research may be needed.

In our research study, we conducted a comprehensive Literature review to determine effective Deep learning algorithms for digit and text recognition. The purpose of the review was to identify the algorithms that have demonstrated high accuracy in object detection tasks and which could potentially be used for our study. Our review revealed that Faster R-CNN, YOLOv5, YOLOv7, and RetinaNet have demonstrated high accuracy in object detection tasks. Specifically, RetinaNet, with its focal loss function, has shown to be effective in object detection. In the paper [74] we reviewed, RetinaNet was used for digit recognition and delivered good accuracy, making it an attractive choice for our study. Many articles and papers have compared the performance of Faster R-CNN and YOLO in object detection, including tests for digit and text recognition. These algorithms were selected based on the findings of our Literature review and will be compared in our study under similar experimental conditions to determine the best deep learning model for handwritten digit dataset [10]. It could be argued that other deep learning algorithms(not mentioned in the review) could also be a potential choice for our study However, based on our search string and the articles obtained we confined our choice of algorithms as mentioned in the LR.

6.1 Analysis on Detection Results

- **Faster R-CNN** : Faster R-CNN from the figure 5.1 we can see that Faster-RCNN could not detect all the digits successfully.Even the detected digits were relatively closer to the threshold.From the figure 5.2 we can see the diagonal of the confusion matrix which represents the accuracies of the individual digits.In that figure 5.2 detection accuracies for the digits "2", "3", "6", "8" are below 80%.
- **RetinaNet** : RetinaNet from the figure 5.3 we can see that RetinaNet clearly

outperforms Faster R-CNN but it clearly misses out on detecting the degraded digits. It is interesting to note that RetinaNet could not detect digit "4" but a part of it was detected as digit "1" which resembles digit "1". From figure 5.4 we can see that RetinaNet outperforms Faster R-CNN on all the digits except digit "5". The detection accuracies for all the digits are well above 80% except the digits "2" and "6" which were closer to 80%.

- **YOLOv5** : YOLOv5 from figure 5.5 we can see that it outperforms Faster R-CNN and RetinaNet and managed to detect most of the digits from the images even the faded and degraded ones. From figure 5.6 we can see that YOLOv5 outperforms RetinaNet and faster RCNN in digits "9", "7", "6", "4", "2", "0".
- **YOLOv7** : YOLOv7 from figure 5.7 we can see that it outperforms the above deep learning models and managed to detect all the digits except in one instance. The detections had a relatively higher confidence score. From figure 5.8 we can see that YOLOv7 has achieved detection accuracies above 90% for "0", "8", "9". As we can see in the figure, it clearly outperforms the above algorithms on all the digits except the digit "5" for which Faster R-CNN has acquired higher accuracy compared to all algorithms.

With efficient deep learning techniques discovered from RQ1, RQ2 is resolved. The DIDA dataset is used to train and test the developed deep learning models, which are then assessed using the accuracy, precision, and recall performance criteria. As shown in Table 5.2, the Accuracy, Precision, and Recall obtained for Faster R-CNN are 79.21%, 70.98%, and 77.3%, correspondingly; for RetinaNet, they are 84.52%, 75.3%, and 77.5%; for YOLOv5, they are 85.7%, 93.21%, and 81.3%; and for YOLOv7, they are 88.7%, 93.7%. The metrics results indicate that the most effective deep learning model is YOLOv7.

6.2 Threats to validity

6.2.1 Internal validity

Internal validity investigates the level of systematic mistakes (bias) [78]. Performance bias, detection bias, and selection bias are all potential sources of this type of systematic inaccuracy. It can occasionally be enhanced if internal validity is compromised. To mitigate the bias, proper algorithms, data, and metrics are considered so that poor performance will not occur at the end of the results.

6.2.2 External validity

Examining a study's external validity determines whether its findings may be applied to different situations [78]. The training and testing data have been taken care of to avoid overfitting. The results obtained in our study cannot be replicated on other datasets as machine learning algorithms are entirely dependent on the data used.

6.2.3 Construct Validity

Construct validity is about how closely the study's findings relate to its concepts or underlying theory [78]. The choice of metrics used to evaluate the performance of object detection models can impact construct validity. Appropriate measures and clearly defining the operational variables mitigate the problems that arise in construct validity.

6.2.4 Conclusion validity

Checking for conclusion validity ensures that research findings are accurate [78]. These risks typically appear when research is carried out improperly and due to the selection of wrong assessment and performance metrics. This study reduces this validity by doing the research according to the right procedures and using the right metrics to assess and compare the deep learning algorithms. The procedures and metrics are considered from the previous papers and discussed with the supervisor.

7.1 Conclusion

The literature review analysis and experimentation identified the YOLOv5, YOLOv7, Faster-RCNN, and RetinaNet algorithms as suitable advanced deep learning algorithms in this study. They were trained using the DIDA dataset, which contains images of ancient handwritten digits with issues such as single-touch, multi-touch, broken, Degradation, and Fainted. Collecting and preserving historical documents is much helpful for the future, so we preprocess the data to remove noise and errors from the images and train the algorithms to detect the digits regardless of how noisy the image might be. The algorithms were evaluated using the metrics such as accuracy, precision, and recall after training them on 13,000 images.

The results of the experiment demonstrate that all algorithms have been successful in detecting the digits. However, all algorithms are compared to determine which approach performs better. From the detection results, we identify that YOLOv7 outperforms the other deep learning models (Faster R-CNN, RetinaNet, and YOLOv5) in detecting degraded digits. It has high detection accuracy for digits "1", "0", "8", and "9". The performance of RetinaNet is better than Faster R-CNN but not as good as YOLOv5 or YOLOv7. Faster R-CNN has a lower detection accuracy compared to the other models, especially for digits "2", "3", "6", and "8". We found that YOLOv7 has the highest accuracy, precision, and recall compared to other algorithms. The accuracy for YOLOv7 is 88.7%, the precision shown is 93.7%, and the recall obtained is 83.9%. These measures demonstrate that the YOLOv7 has a higher score and is more effective at detecting and recognizing handwritten digits.

7.2 Future Work

In this thesis, we have worked on detecting and recognizing handwritten digits using four algorithms. The thesis's further work may focus on enhancing the accuracy and effectiveness of present methods. Every algorithm can have an improved version by using different hyperparameters or sources. Future work can also be focused on implementing the models regarding the digit issues addressed in this thesis. Future work can include exploring and expanding the other different use cases with these algorithms.

References

- [1] E. Granell, E. Chammas, L.L. Sulem, C.D.M. Hinarejos, C. Mokbel, B.I. Cirstea, Transcription of Spanish historical handwritten documents with deep neural networks, *J. Imaging* 4 (15) (2018) 1–22.
- [2] D. Dahlström and B. Boström, "Pros and Cons: Handwriting Versus Digital Writing," *Nordic Journal of Digital Literacy*, vol. 12, (4), pp. 143-161, 2017.
- [3] O. Elitez, Handwritten digit string segmentation and recognition using deep learning, Master's thesis, Middle East Technical University, Turkey, 2015.
- [4] Sarker IH. Ai-driven cybersecurity: an overview, security intelligence modeling and research directions. *SN Comput Sci.* 2021.
- [5] SuperAnnotate, "Introduction to object detection with deep learning," SuperAnnotate Blog, 19-Oct-2021. [Online]. Available: <https://blog.superannotate.com/object-detection-with-deep-learning/>. [Accessed: 06-Feb-2022].
- [6] Steinkraus, D. et al. (2005). Using GPUs for machine learning algorithms. Proceedings. Eighth International Conference on Document Analysis and Recognition (ICDAR'05) (pp. 1115-1120), 2005
- [7] Y. LeCun, C. Cortes, C.J. Burges, MNIST handwritten digit database, <http://yann.lecun.com/exdb/mnist>, 2010.
- [8] P.J. Grother, NIST special database 19, 2017, <https://www.nist.gov/srd/nist-special-database-19>.
- [9] J.J. Hull, A database for handwritten text recognition research, *IEEE Trans. Pattern Anal. Mach. Intell.* 16 (5) (1994) 550–554.
- [10] Huseyin Kusetogullari, Amir Yavariabdi, Johan Hall, Niklas Lavesson, DIDA: The largest historical handwritten digit dataset with 250k digits, June 2021. Accessed on: June 13, 2021. Available: <https://github.com/didadataset/DIDA/>.
- [11] Kusetogullari, H., Yavariabdi, A., Cheddad, A. et al. ARDIS: a Swedish historical handwritten digit dataset. *Neural Comput & Applic* 32, 16505–16518 (2020). <https://doi.org/10.1007/s00521-019-04163-3>
- [12] Cheddad, A., Kusetogullari, H., Hilmkil, A. et al. SHIBR—The Swedish Historical Birth Records: a semi-annotated dataset. *Neural Comput & Applic* 33, 15863–15875 (2021). <https://doi.org/10.1007/s00521-021-06207-z>
- [13] M. Middleton, "Deep Learning vs. Machine Learning — What's the Difference?," Flatiron School, Feb. 08, 2021. <https://flatironschool.com/blog/deep-learning-vs-machine-learning/>

- [14] Wakefield, K. (n.d.). A guide to the types of machine learning algorithms. SAS. <https://www.sas.com/engb/insights/articles/analytics/machine-learning-algorithms.html>
- [15] Marr, B. (2021, July 13). How Do You Know When And Where To Apply Deep Learning? Bernard Marr. <https://bernardmarr.com/how-do-you-know-when-and-where-to-apply-deep-learning/>
- [16] Lohia, Aditya; Kadam, Kalyani Dhananjay; Joshi, Rahul Raghvendra; and Bongale, Dr. Anupkumar M., "Bibliometric Analysis of One-stage and Two-stage Object Detection" (2021). Library Philosophy and Practice (e-journal). 4910.
- [17] Z. -Q. Zhao, P. Zheng, S. -T. Xu and X. Wu, "Object Detection With Deep Learning: A Review," in IEEE Transactions on Neural Networks and Learning Systems, vol. 30, no. 11, pp. 3212-3232, Nov. 2019, doi: 10.1109/TNNLS.2018.2876865.
- [18] Ren, J. and Wang, Y. (2022) Overview of Object Detection Algorithms Using Convolutional Neural Networks. Journal of Computer and Communications, 10, 115-132.
- [19] R. Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation," 2013.
- [20] Padaliya, V. (2021, December 12). Difference between AI, ML and DL - Decoding Artificial Intelligence. Medium. <https://medium.com/decoding-artificial-intelligence/difference-between-ai-ml-and-dl-f5274d8db2>
- [21] Deep Learning for Ligand-Based Virtual Screening in Drug Discovery-Scientific Figure on ResearchGate. Available from:https://www.researchgate.net/figure/Deep-Neural-Network-architecture_fig1_33012003
- [22] How RetinaNet works? | ArcGIS API for Python. (n.d.). <https://developers.arcgis.com/python/guide/how-retinanet-works/>
- [23] Complexity and accuracy analysis of common artificial neural networks on pedestrian detection - Scientific Figure on ResearchGate. Available from:https://www.researchgate.net/figure/YOLO-architecture-YOLO-architecture-is-inspired-by-GooLeNet-model-for-image_fig2_329038564 [accessed 6 Jan, 2023]
- [24] Towards AI Editorial Team. (2022, January 7). Training Faster R-CNN Using TensorFlow's Object Detection API with a Custom Dataset. Medium. <https://pub.towardsai.net/training-faster-r-cnn-using-tensorflow-object-detection-api-with-a-custom-dataset-88dd525666fd>
- [25] V. Rajput, "Yolov7: Making YOLO Great Again," AIGuys, Nov. 24, 2022. <https://medium.com/aiguys/YOLOv7-making-YOLO-great-again-7b1ec1f6a2a0>.
- [26] A Forest Fire Detection System Based on Ensemble Learning - Scientific Figure on ResearchGate. Available from: <https://www.researchgate.net/figure/The>

- network-architecture-of-YOLOv5-It-consists-of-three-parts-1-Backbone-CSPDarknet_fig1_349299852
- [27] Park, S. (2022, January 6). A guide to Two-stage Object Detection: R-CNN, FPN, Mask R-CNN. Medium. <https://medium.com/codex/a-guide-to-two-stage-object-detection-r-cnn-fpn-mask-r-cnn-and-more-54c2e168438c>
- [28] X. Sun, P. Wu and S. C. H. Hoi, "Face detection using deep learning: An improved Faster R-CNN approach," *Neurocomputing (Amsterdam)*, vol. 299, pp. 42-50, 2018.
- [29] Gad, A. F. (2021, April 9). Faster R-CNN Explained for Object Detection Tasks. Paperspace Blog. <https://blog.paperspace.com/faster-r-cnn-explained-object-detection/>
- [30] T. Lin et al, "Focal Loss for Dense Object Detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, (2), pp. 318-327, 2020.
- [31] "How RetinaNet works? | ArcGIS Developer," [developers.arcgis.com. https://developers.arcgis.com/python/guide/how-retinanet-works/](https://developers.arcgis.com/python/guide/how-retinanet-works/)
- [32] Karimi, G. (n.d.). Introduction to YOLO Algorithm for Object Detection. Section. <https://www.section.io/engineering-education/introduction-to-YOLO-algorithm-for-object-detection/>
- [33] Solawetz, J. (2022). YOLOv4. roboflow. <https://blog.roboflow.com/a-thorough-breakdown-of-YOLOv4/#:~:text=In%20summary%2C%20YOLOv4%20is%20a,and%20efficiently%20for%20object%20detection.>
- [34] D. Snegireva and A. Perkova, "Traffic Sign Recognition Application Using YOLOv5 Architecture," 2021 International Russian Automation Conference (RusAutoCon), 2021, pp. 1002-1007, doi: 10.1109/RusAutoCon52004.2021.9537355.
- [35] Xu, Renjie Lin, Haifeng Lu, Kangjie Cao, Lin Liu, Yunfei. (2021). A Forest Fire Detection System Based on Ensemble Learning. *Forests*. 12. 217. 10.3390/f12020217
- [36] C. Wang, A. Bochkovskiy and H. M. Liao, "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," 2022.
- [37] S. Pashine, R. Dixit and R. Kushwah, "Handwritten Digit Recognition using Machine and Deep Learning Algorithms," 2021.
- [38] R. Sethi and I. Kaushik, "Hand written digit recognition using machine learning," in 2020, . DOI: 10.1109/CSNT48778.2020.9115746.
- [39] Sahu, Vijay Laxmi, and Babita Kubde. "Offline handwritten character recognition techniques using neural network: a review." *International journal of science and Research (IJSR)* 2.1 (2013): 87-94.
- [40] B. M. Vinjit, M. K. Bhojak, S. Kumar and G. Chalak, "A Review on Handwritten Character Recognition Methods and Techniques," 2020 International Conference on Communication and Signal Processing (ICCSP), 2020, pp. 1224-1228, doi: 10.1109/ICCSP48568.2020.9182129.

- [41] C. Boufenar, A. Kerboua and M. Batouche, "Investigation on deep learning for off-line handwritten Arabic character recognition," *Cognitive Systems Research*, vol. 50, pp. 180-195, 2018.
- [42] Xiaolin Li, R. Plamondon and M. Parizeau, "Model-based online handwritten digit recognition," *Proceedings. Fourteenth International Conference on Pattern Recognition (Cat. No.98EX170)*, Brisbane, Queensland, Australia, 1998, pp. 1134-1136 vol.2, doi: 10.1109/ICPR.1998.711895.
- [43] H. Zhan, S. Lyu and Y. Lu, "Handwritten Digit String Recognition using Convolutional Neural Network," *2018 24th International Conference on Pattern Recognition (ICPR)*, 2018, pp. 3729-3734, doi: 10.1109/ICPR.2018.8546100.
- [44] A. Chakraborty, R. De, S. Malakar, F. Schwenker and R. Sarkar, "Handwritten Digit String Recognition using Deep Autoencoder based Segmentation and ResNet based Recognition Approach," *2020 25th International Conference on Pattern Recognition (ICPR)*, 2021, pp. 7737-7742, doi: 10.1109/ICPR48806.2021.9412198.
- [45] S. Aly and A. Mohamed, "Unknown-Length Handwritten Numeral String Recognition Using Cascade of PCA-SVMNet Classifiers," *IEEE Access*, vol. 7, pp. 52024-52034, 2019.
- [46] De Sousa Neto, Arthur Flor et al, "HDSR-Flor: A Robust End-to-End System to Solve the Handwritten Digit String Recognition Problem in Real Complex Scenarios," *IEEE Access*, vol. 8, pp. 208543-208553, 2020.
- [47] H. Kusetogullari, H. Grahn and N. Lavesson, "Handwriting image enhancement using local learning windowing, Gaussian Mixture Model and k-means clustering," *2016 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT)*, 2016, pp. 305-310, doi: 10.1109/ISSPIT.2016.7886054.
- [48] Madakannu, A., Selvaraj, A. DIGI-Net: a deep convolutional neural network for multi-format digit recognition. *Neural Comput & Applic* 32, 11373–11383 (2020). <https://doi.org/10.1007/s00521-019-04632-9>.
- [49] S. Ali et al, "An efficient and improved scheme for handwritten digit recognition based on convolutional neural network," *SN Applied Sciences*, vol. 1, (9), pp. 1-9, 2019.
- [50] S. Tabik et al, "A snapshot of image pre-processing for convolutional neural networks: case study of MNIST," *International Journal of Computational Intelligence Systems*, vol. 10, (1), pp. 555, 2017.
- [51] A. Cheddad, H. Kusetogullari and H. Grahn, "Object recognition using shape growth pattern," *Proceedings of the 10th International Symposium on Image and Signal Processing and Analysis*, 2017, pp. 47-52, doi: 10.1109/ISPA.2017.8073567.
- [52] A. Challa, 'Automatic Handwritten Digit Recognition On Document Images Using Machine Learning Methods', Dissertation, 2019.
- [53] M. Zhao, 'Handwritten digit recognition based on segmentation-free method', Dissertation, 2020.

- [54] 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, 2021.
- [55] Claes Wohlin, Per Runeson, Martin Höst, Magnus C Ohlsson, Björn Regnell, and Anders Wesslén. *Experimentation in Software Engineering*. Springer, Germany, 2012.
- [56] Per Runeson and Martin Höst. Guidelines for conducting and reporting case study research in software engineering. *Empirical software engineering : an international journal*, 14(2):131–164, 2008;2009;.
- [57] G. Bradski, “The OpenCV Library,” *Dr. Dobb’s Journal of Software Tools*, 2000.
- [58] W. McKinney et al., “pandas: a foundational python library for data analysis and statistics,” *Python for high performance and scientific computing*, vol. 14, no. 9, pp. 1–9, 2011.
- [59] F. Fabris and A. A. Freitas, *Analysing the Overfit of the Auto-sklearn Automated Machine Learning Tool*, vol. 11943 of *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, pp. 508–520. Cham: Springer International Publishing, 2020;2019;.
- [60] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, “Array programming with numpy,” 2020.
- [61] J. D. Hunter, “Matplotlib: A 2d graphics environment,” *Computing in science engineering*, vol. 9, no. 3, pp. 90–95, 2007.
- [62] A. Maddalone, M. Moocarme and A. So, *The TensorFlow Workshop*. 2021.
- [63] B. Ramsundar and R. B. Zadeh, *TensorFlow for Deep Learning*. 2018.
- [64] L. P. G. Antiga, E. Stevens and T. Viehmann, *Deep Learning with Pytorch*. 2020.
- [65] A. Gulli and S. Pal, *Deep Learning with Keras: Implement Neural Networks with Keras on Theano and TensorFlow*. (1st ed.) 2017.
- [66] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.
- [67] A. Geron, *Hands-on Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. 2017.
- [68] Faizan, Muhammad. “Roboflow - Red Buffer.” *Medium*, 16 Apr. 2022, medium.com/red-buffer/roboflow-d4e8c4b52515.
- [69] Albahli, S., Nawaz, M., Javed, A. et al. An improved faster-RCNN model for handwritten character recognition. *Arab J Sci Eng* 46, 8509–8523 (2021). <https://doi-org.miman.bib.bth.se/10.1007/s13369-021-05471-4>

- [70] W. Zhiming, W. Wuxi and X. Yuxiang, "Automatic Container Code Recognition via Faster-RCNN," 2019 5th International Conference on Control, Automation and Robotics (ICCAR), 2019, pp. 870-874, doi: 10.1109/ICCAR.2019.8813401.
- [71] Raducu Gavrilesu et al. "Faster R-CNN: an Approach to Real-Time Object Detection". In: 2018 International Conference and Exposition on Electrical And Power Engineering (EPE). IEEE. 2018, pp. 0165–0168.
- [72] Hendry and R. Chen, "Automatic License Plate Recognition via sliding-window darknet-YOLO deep learning," *Image and Vision Computing*, vol. 87, pp. 47-56, 2019.
- [73] K. Patel, C. Bhatt and P. L. Mazzeo, "Improved Ship Detection Algorithm from Satellite Images Using YOLOv7 and Graph Neural Network," *Algorithms*, vol. 15, (12), pp. 473, 2022.
- [74] A. G. Hochuli et al, "A comprehensive comparison of end-to-end approaches for handwritten digit string recognition," *Expert Systems with Applications*, vol. 165, pp. 114196, 2021.
- [75] N. Annapareddy et al, "Handwritten text and digit classification on rwandan perioperative flowsheets via YOLOv5," in 2022, . DOI: 10.1109/SIEDS55548.2022.9799426.
- [76] C. Coelho et al, "Object detection with RetinaNet on aerial imagery: The algarve landscape," in *Computational Science and its Applications – ICCSA 2021* Anonymous Cham: Springer International Publishing, 2021, pp. 501-516.
- [77] T. M. Hoang et al, "Deep RetinaNet-Based Detection and Classification of Road Markings by Visible Light Camera Sensors," *Sensors (Basel, Switzerland)*, vol. 19, (2), pp. 281, 2019.
- [78] D. Budgen, P. Brereton and B. A. Kitchenham, *Evidence-Based Software Engineering and Systematic Reviews*. 2015. DOI: 10.1201/b19467.

Appendix A

Supplemental Information

```
1_jpg.rf.5c294c657b6a2bde8bf3b86cc7e3313.txt
6 0.828125 0.4879807692307692 0.2578125 0.6448317307692307
3 0.5853365384615384 0.5264423076923077 0.21875 0.6502403846153846
8 0.3786057692307692 0.5108173076923077 0.22926682692307693 0.5985576923076923
1 0.20793269230769232 0.5492788461538461 0.19951923076923078 0.6616586538461539
```

Figure A.1: Annotation file for YOLOv5 and YOLOv7 in text format

File Path	59	172	187	585	1
140_jpg.rf.4b8131b16797f062e1e40ff535ff0410.jpg	59	172	187	585	1
140_jpg.rf.4b8131b16797f062e1e40ff535ff0410.jpg	170	178	311	591	8
140_jpg.rf.4b8131b16797f062e1e40ff535ff0410.jpg	318	172	438	530	0
140_jpg.rf.4b8131b16797f062e1e40ff535ff0410.jpg	413	150	609	525	5
575_jpg.rf.5c3cfb52878d2e5e2d2f6ad99f76e0aa.jpg	88	114	215	405	1
575_jpg.rf.5c3cfb52878d2e5e2d2f6ad99f76e0aa.jpg	208	119	322	425	8
575_jpg.rf.5c3cfb52878d2e5e2d2f6ad99f76e0aa.jpg	304	146	414	419	1
575_jpg.rf.5c3cfb52878d2e5e2d2f6ad99f76e0aa.jpg	373	140	524	592	9
136_jpg.rf.9bdbfc8dfbe5fd687f5be3f98ed83b52.jpg	21	188	158	608	1
136_jpg.rf.9bdbfc8dfbe5fd687f5be3f98ed83b52.jpg	145	160	261	565	8
136_jpg.rf.9bdbfc8dfbe5fd687f5be3f98ed83b52.jpg	266	190	381	539	0
136_jpg.rf.9bdbfc8dfbe5fd687f5be3f98ed83b52.jpg	378	149	571	535	5
141_jpg.rf.c1da83501d65ffad25277a5995898520.jpg	99	33	202	421	1
141_jpg.rf.c1da83501d65ffad25277a5995898520.jpg	217	2	330	490	8
141_jpg.rf.c1da83501d65ffad25277a5995898520.jpg	347	31	468	458	0
141_jpg.rf.c1da83501d65ffad25277a5995898520.jpg	473	79	601	474	5
1918_jpg.rf.db4e38e11159490fc3201098cfc09cac.jpg	116	131	253	372	1
1918_jpg.rf.db4e38e11159490fc3201098cfc09cac.jpg	232	127	357	346	8
1918_jpg.rf.db4e38e11159490fc3201098cfc09cac.jpg	326	129	471	337	2
1918_jpg.rf.db4e38e11159490fc3201098cfc09cac.jpg	133	121	579	514	9
266_jpg.rf.5fa8063a7e87c9bb43c17f62d66e8f27.jpg	114	183	322	500	7
266_jpg.rf.5fa8063a7e87c9bb43c17f62d66e8f27.jpg	249	96	539	514	5

Figure A.2: Annotation file for Retinanet in .csv format

```

feature {
  key: "image/object/bbox/ymax"
  value {
    float_list {
      value: 0.8656250238418579
      value: 0.8671875
      value: 0.846875011920929
      value: 0.9140625
    }
  }
}
feature {
  key: "image/object/bbox/ymin"
  value {
    float_list {
      value: 0.4703125059604645
      value: 0.4703125059604645
      value: 0.41718751192092896
      value: 0.40312498807907104
    }
  }
}
feature {
  key: "image/object/class/label"
  value {
    int64_list {
      value: 2
      value: 7
      value: 2
      value: 8
    }
  }
}
feature {
  key: "image/object/class/text"
  value {
    bytes_list {
      value: "1"
      value: "8"
      value: "1"
      value: "9"
    }
  }
}
feature {
  key: "image/width"
  value {
    int64_list {
      value: 640
    }
  }
}

```

Figure A.3: Annotation file for Faster R-CNN in .tfrecord format.

