

Real-Time Garments Defects Detection at the sewing phase
to optimize waste cost using YOLOv7, YOLOv7x,
YOLOv7-w6 and Pytorch

by

Md. Minhaz Uddin

19101013

Sadi Mahmud Foysal

22241046

Sadia Rahman

19141001

Nushara Tazrin Risti

22241041

Sanzeda Akter Sarmin

19101026

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science

Department of Computer Science and Engineering
Brac University
January 2023

© 2023. Brac University
All rights reserved.

Declaration

It is hereby declared that

1. The thesis submitted is our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material that has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:



Md. Minhaz Uddin
19101013



Sadi Mahmud Foysal
22241046



Sadia Rahman
19141001



Nushara Tazrin Risty
22241041



Sanzeda Akter Sarmin
19101026

Approval

The thesis/project titled “Real-Time Garments Defects Detection at the sewing phase to optimize waste cost using YOLOv7, YOLOv7x, YOLOv7-w6 and Pytorch” submitted by

1. Md. Minhaz Uddin (19101013)
2. Sadi Mahmud Foysal (22241046)
3. Nushara Tazrin Risty (22241041)
4. Sanzeda Akter Sarmin (19101026)
5. Sadia Rahman (19141001)

Of Fall, 2022 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on January 17, 2023.

Examining Committee:

Supervisor:
(Member)



Dr. Md. Khalilur Rhaman
Professor
Department of Computer Science and Engineering
Brac University

Program Coordinator:
(Member)

Dr. Md. Golam Rabiul Alam
Professor
Department of Computer Science and Engineering
Brac University

Head of Department:
(Chair)

Dr. Sadia Hamid Kazi
Associate Professor
Department of Computer Science and Engineering
Brac University

Abstract

In the era of computer vision to overcome challenges, the introduction of the YOLO model revolutionized real-time computer vision approaches. In the garment industry, the inception of products plays a significant role while increasing the processing time with a good accuracy rate is the big challenge here. A real-time garments defect detection approach using YOLOv7, YOLOv7x, and YOLOv7-w6 on a primary dataset is proposed with a good FPS rate and better accuracy. Maximum traditional garments inception approaches focused on end product defects while this model suggests detecting defects on the sewing phase so that the cost of the rejected end product can be optimized by detecting them before a product goes through all the phases. For this our research is more focused on three subclasses of Seam, Stitch, and Hole related to sewing phase defects. To increase the detection rate, the hyperparameter tuning technique is applied to the YOLOv7 model. Three models are proposed based on pre-trained weights of YOLOv7, YOLOv7x, and YOLOv7-w6 to compare the accuracy and FPS rate in terms of implementation in real-world projects.

Keywords: TILDA, YOLOv7, YOLOv7x, YOLOv7-W6 CNN, RCNN, DCNN, Faster-RCNN, Neural Networks, Real-Time, computer vision, fault, Defect, Detection, Garments, Hole, Stitch, Seam, Roboflow,

Dedication

We dedicated this thesis to our parents.

Acknowledgement

Firstly, all praise to the Great Allah for whom our thesis have been completed without any major interruption.

Secondly, to our Advisor Dr. Md. Khalilur Rhaman sir for his kind support and advice in our work. He helped us whenever we needed help.

Table of Contents

Declaration	i
Approval	ii
Abstract	iii
Dedication	iv
Acknowledgment	v
Table of Contents	vi
Nomenclature	viii
1 Introduction	1
1.1 Overview	1
1.2 Research Problem	2
1.3 Research Objectives	2
1.4 Research Orientation	3
2 Literature Review	4
3 Work Plan	7
4 Object Detection and YOLO	8
4.1 Object Detection and Its types	8
4.2 YOLO	9
4.3 Pytorch	11
5 Methodology	13
5.1 Dataset	13
5.1.1 Process of Data collection	13
5.1.2 Hassle of Data Collection	13
5.1.3 Data annotation	15
5.1.4 Labeled Classes	16
5.1.5 Train-Valid-Test Split	18
5.1.6 Data analysis	18
5.1.7 YOLOv7	19
5.2 YOLOv7 Architecture	20
5.3 Model Description	22

6	Model Training	25
6.1	Yolov7	25
6.1.1	mAP@.5=0.39	26
6.1.2	mAP@0.5=0.602	27
6.2	YOLOv7x	29
6.3	YOLOv7-w6	32
7	Result Analysis	36
7.1	Result of YOLOv7	36
7.2	Result of YOLOv7x	37
7.3	Result of YOLOv7-w6	39
7.4	Real-time FPS calculation	40
7.4.1	Comparison of Results	41
8	Conclusion	42
8.1	Conclusion	42
8.2	Future Scope	42
	bibliography	44

Nomenclature

The next list describes several symbols & abbreviation that will be later used within the body of the document

AP Average Precision

CNN Convolutional Neural Network

CUDA Compute Unified Device Architecture

FPS Frames Per Second

GPU Graphics Processing Unit

IOU Intersection over Union

mAP Mean Average Precision

QC Quality Check

RMG Ready Made Garments

RPN Region Proposal Network

SGD Stochastic Gradient Descent

YOLO You Only Look Once

Chapter 1

Introduction

1.1 Overview

Since its inception in the late 1970s, Bangladesh's garment industry has been a significant contributor to the country's economic development, growing to become the country's top export-generating industry. Bangladesh is a developing nation, thus the RMG industry has evolved into one of the nation's lifelines and generates a significant share of its export earnings. This company has improved socioeconomic prospects by offering many work opportunities, especially to the nation's uneducated and underpaid female labor force. As a consequence, the RMG industry has significantly contributed to women's empowerment by creating employment. So, to maintain this industry up to date, it is important to manufacture good, high-quality products. It's very challenging to manufacture a fault-free product, even with good raw ingredients, labor, and fabric. Because the product's value was diminished owing to a flaw in it, for this reason, the garments product must be checked before they are offered to consumers. Traditionally, competent employees use visual sense to assess flaws. The manual detection approach has drawbacks such as a high mistake rate due to human weariness and slow execution (experienced personnel can only identify 15–20 per minute)[3]. As a result, in order to improve productivity, an automated detecting system must be developed. There are different kinds of automated detecting systems and processes that have been developed throughout the ages. This algorithm is mainly differing in terms of the flaws like fabric or different parts of the product as well. As like Pattern recognition is done via cross-correlation. With a multi-layer artificial neural network, Campell et al. employed the discrete Fourier transform to detect errors in denim fabric. Faults are recognized using collected features in a maximum probability classifier using the Fourier transform. Used the Fourier transform-based spherical image refinement scheme to detect faults in Product fabric[8]. There are many more such kinds of algorithms some are costly some are not much efficient to detect every fault some are not portable to reach every section of the garment product to check whether there is a fault or not. The largest problem is real-time defect detection, and to detect faults like stitching, holes, and yarn faults, which are getting smaller and smaller, we need a better algorithm. This study suggests a deep learning approach to detect fabric defects automatically. An enhanced YOLOv7 object recognition technique for fabric flaws is suggested to address the issues of uneven forms and many little objects. to increase the tiny item detection's precision.

1.2 Research Problem

This system's primary goal is to quickly and accurately identify fabric flaws, such as small holes, stitching issues, and cloth defects. This implies that we will be able to identify the flaw in a matter of seconds for which YOLOv7 was applied. After training the model, we next utilized a portable device that was connected to the sewing machine so that, while sewing, we could check for stitching errors. We could also use this portable device manually to check for errors in fabrics with a lot of intricate folds. The ready-made garments (RMG) business in Bangladesh is the most varied in terms of growth and foreign currency gains. It substantially strengthens the nation's economy by creating a large number of job possibilities and reducing poverty via socio-economic development. The RMG sector's share of the country's exports has grown dramatically over time. It has evolved into Bangladesh's primary export sector. Over the last 20 years, the RMG sector has emerged as the country's development and expansion engine. The export revenues of the garment sector climbed from 1 million in 1978 to 31.57 million in 1983–1984 to 10.7 billion in 2007 to 24.49 billion in 2013–2014, accounting for 81.13 percent of the nation's overall export revenue.[1]. As a result, in order to maintain these states and increase this proportion, the efficiency in this sector must be increased. Because quality control is one of the most important steps in the production of RMG goods and takes a significant amount of time and labor, lowering these three costs is the top priority. If the procedure is automated by a machine, checking will take less time overall. By employing less labor in this area, we may use that labor in manufacturing, increasing our earnings from the previous period. Humans can only work continuously for up to 6 hours before they need a break, and it can take them almost 3 to 5 minutes to check a product. If this process is automated with the aid of a handheld device, the time could be cut in half or the machine could perform tasks more quickly than a human, saving time and improving the effectiveness of the work.

1.3 Research Objectives

The major objective of this research is to reduce defects, particularly holes, stitches, and seams, in the quality of ready-made garments and to improve productivity through real-time detection, which will reduce production time, and labor costs, and work will be done more efficiently, and the buyer will receive much better quality products.

1. Need to train model through fault data so that it can understand which are defects and which are defectless cloth.
2. Need to implement it to a portable device and need to attach it to a sewing machine so that faults could be recognized.
3. Real-time detection needs to be done for efficiency and to reduce the time to find the defects in the QC or to get the fault before QC.

1.4 Research Orientation

1. Most of the data were produced by the team members as we collected the image of defects of our own clothes and also collected data from different platforms we visited some garments to get the defects and then inquire as to what type of fault they needed to deal with we have also collected the least amount of data from the online platform.
2. Through the help of the website Roboflow we were able to annotate and augmented our data. To get the best accuracy with your different augmentation processes.
3. To begin, the data were trained into a CNN model to achieve accuracy, and then the model YOLOv7 was used for real-time detection.

Chapter 2

Literature Review

Deep Learning has been shown in recent years to outperform traditional computer vision algorithms in a wide range of computer vision applications. Deep learning models improved processing speed as well as accuracy. "TRUE" Because accuracy is critical, real-time object detectors must run at least 30 frames per second, though slower models are also considered. Slower detectors are more accurate is a widely held belief that may be classified as a half-truth because it is not always true[19]. The majority of deep learning-based object detection methods employ one- and two-stage object identification algorithms. The suggested areas are first retrieved from the network's input image, and each suggested region is then subjected to classification and location regression. Object detection is completed in two steps using this two-stage technique. Following that, object detection is activated. R-CNN[5, SPP-Net, Fast RCNN[7, and Faster R-CNN] are among these algorithms[9]. Instead of requiring an area proposal step, the one-stage object detection approach immediately produces the object's simulated results and position data points. Among these algorithms are YOLO, SSD, YOLOv2, YOLOv3, RetinaNet, YOLOv4, and YOLOv5[14]. The RCNN is a fast and flexible framework for defect detection. It outperformed the most recent detection algorithm available at the time of publication by more than 30% on the PASCAL VOC 2012 dataset with an accuracy of 53.3% and the ILSVRC2013 detection dataset with a precision of 31.4. Two significant discoveries were used in the approach to accomplishing this feat. Using bottom-up area proposals (extracted earlier; in this case, around 2000), objects are first located and segmented, and then the characteristics for each suggestion are calculated. SPPNET, a cutting-edge deep convolutional neural network framework, focuses on object recognition. This paradigm was inspired by the fact that older frameworks had an arbitrary size restriction on input photographs. Fast-RCNN, which was tested on the PASCAL VOC 2012 dataset, speeds up the training and detection procedures. The VGG16 network trains 9 times faster than the RCNN, detects 213 times faster and has a higher mAP (average classification accuracy). The main idea behind this development is to optimize by considering not only accuracy as well as other factors such as speed. Faster-RCNN is an incremental enhancement to the Region-based CNN architecture. According to the new technique, the previous method's bottleneck (where the algorithm was slow) and not the CNN computations were to blame for the "lack" of speed. YOLO detects objects in real-time while the detector runs at 45 frames at a time. At 155 frames per second, a smaller detecting network is active. In terms of accuracy/speed trade-off, the full-size detector is the

most effective option for object detection. It was decided to use a new strategy because previous detectors used classifiers as detectors, making it more difficult to adjust for accuracy and performance. This novel method views object recognition as a regression issue to bounding boxes that are separated in space and class probabilities. In this method, one network directly predicts both the bounding boxes and the probabilities, resulting in a single evaluation[2]. This enabled significant speed gains by accounting for direct detection performance and making overall process optimization easier. The YOLOv2 detector is an enhanced version of the original YOLO detector that still outperforms the competition on well-known detection datasets like PASCAL VOC and COCO. It still has a speed advantage over the competition. The YOLOv2 model can run on the PASCAL VOC 2007 dataset at various trade-offs using the multi-scale training method they provided (which trains both on proper detection and classification)[6]. They are respectively 76.8 mAP at 67 FPS and 78.6% at 40 FPS. Several of the original YOLO's flaws have been cleverly fixed in YOLOv2, including In addition to increasing resolution so that YOLOv2 trains at 448x448 pixel density on ImageNet for 10 epochs, and the filters are able to adapt to high-resolution inputs, batch normalization in each convolutional layer also helps with better convergence. The third version of the YOLO approach is known as YOLOv3. It is presented as a technical update or report and incorporates a few minor changes to the original YOLO method[10]. With these adjustments, they were able to run 3–8 times as rapidly and retain real-time detection while still achieving accuracy that was equivalent to that of cutting-edge techniques. It performs 28.2 mAP in 22 ms at 320x320 image size, whereas SSD performs 31.2 mAP in 125 ms. The modern RetinaNet-101-800 operates in less than 200 milliseconds at 37.8 mAP (9–10 times slower). YOLOv4 was the most recent version of the YOLO one-stage detection framework. The primary goal was to create a detector that was not only easier to train and use in real-time, but also more reliable and accurate[2]. On PASCAL VOC 2012, the performance is superb, with 38% mAP at 120 FPS or 43% mAP at 80 FPS. This enables very precise real-time detection. These kinds of advancements are required for self-driving cars and real-time networked industries. MS COCO accuracy on a Tesla V100 was 43.5% mAP and 65.7% AP50 at 65 FPS. YOLOv5 uses the top convolutional neural network optimization techniques created by Glenn Jocher in the year 2020 and is based on the original YOLO object identification architecture[13]. To extract important information from input samples, the YOLOv5 backbone network employs the Focus structure in conjunction with the CSP structure. Yolov5 delivers predictions on three different feature layer sizes by connecting the feature layers using the PANet framework. The YOLOv5 architecture is made up of the architectures YOLOv5s, YOLOv5m, YOLOv5l, and YOLOv5x[14]. The main distinction between them is the number of feature extraction modules and convolution kernels present in each specific area of the network. As a result, the research improved the design of the fabric object recognition network based on the YOLOv5s architecture by carefully considering the identification model's accuracy, efficacy, and size. The use of traditional machine learning techniques is somewhat limited. To address this issue, the proposed technique employs a three-layered DCNN for the representation of the normal and faulty fabric patch. The proposed DCNN's performance is evaluated using % accuracy on the internal database and standard TILDA. The accuracy of the suggested technique is 99.06% for the non-patterned TILDA database and 98.33 and 90.39%, respectively, for pat-

terned and non-patterned fabric defect identification[11]. Although the accuracy of YOLO and Fast-RCNN may be comparable, YOLO is quicker in this situation. Two-stage detectors often function better than single-stage detectors[5]. YOLO with single-stage detectors outperformed other methods in object identification due to architectural successors and detection accuracy. To boost the speed of defect detection, an improved YOLOv3 detection layer is introduced. The results of the experiments show that the proposed method efficiently lowers the network’s error rate and has applications in both lattice and grey cloth networks. Our further study will concentrate on enhancing the method’s real-time performance. YOLOv6 is a single-stage object detection model based on the YOLO architecture (also known as MT-YOLOv6). The YOLOv6 model was produced by Meituan researchers[17]. On the MS COCO dataset, YOLOv6 outperforms YOLOv5 in terms of performance. Afterward An Industrial Application Framework for Single-Stage Object Detection On the COCO data set, YOLOv6-N on an NVIDIA Tesla T4 GPU obtained 35.9% AP at a throughput of 1234 FPS. YOLOv6-S outperforms other major detectors on the same scale, achieving 43.5% AP at 495 FPS (YOLOv5-S, YOLOX-S, and PPYOLOE-S)[15]. The upgraded quantized version of YOLOv6-S also introduced a brand-new, cutting-edge 43.3% AP at 869 FPS. Furthermore, YOLOv6-M/L outperformed other detectors with comparable inference speeds in terms of accuracy (49.5%/52.3%). Researchers meticulously carried out experiments to verify the efficacy of each component[21]. The YOLOv7 model is suggested to identify tiny faults in the setting of a complex backdrop to enhance detection outcomes. In order to create more appropriate anchor boxes for identifying targets with insulator defects, the target boxes of the insulator dataset are first clustered using K-means++. The network is then expanded using the Coordinate Attention (CoordAtt) and Hor-Block modules. The network may then make the feature-extraction process’ useful characteristics stronger while making the undesirable features in the channel and spatial domains weaker. Finally, the positive-negative sample imbalance is resolved and model convergence is sped up by using the SCYLLA-IoU (SIoU) and focused loss functions. Additionally, the non-maximum suppression (NMS) approach is enhanced to decrease unintentional deletion and incorrect detection of defect targets in order to enhance the model’s overall performance[22].

Chapter 3

Work Plan

To detect flaws in garments, the proposed system employs an optical instrument. In order to do this, the model necessitates the creation of a procedure that accepts picture data from an optical device as input, analyzes the data methodically, and produces the required outcome. The model for detecting defects (Figure 3.1) is built using a data set and image processing techniques applied allegorically. The process of recognizing and emphasising various aspects of an image is known as image labelling. When creating metadata for images, it is beneficial to automate the process. To teach the algorithm how to recognize defects, a well-known deep learning method will be used. To begin labelling the images, upload all of the raw images to the system. Following that, image labelling software is installed in order

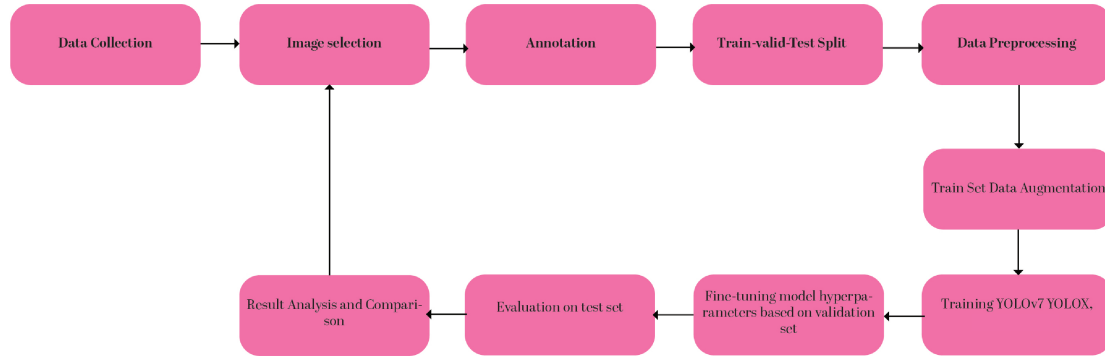


Figure 3.1: Work Plan

to annotate such photos using specific techniques based on the customised criteria. Video data is a common asset that is used every day, whether it is a live stream in a security camera or dash-cam, live video broadcasts are quite simple to access, and we only need to merge a small amount of code for live video access with our garment defect detection.

Chapter 4

Object Detection and YOLO

4.1 Object Detection and Its types

The object detection mechanism in computer vision is used to identify and localize items within an image or video. In object detection, a refined type of image classification, a neural network predicts and identifies items in a picture using bounding boxes. Identifying and locating objects in an image that fit into one of several predefined classes is referred to as "object detection." Due to the numerous real-world applications of tasks like detection, recognition, and localization, object detection, also referred to as object recognition, is an essential subfield in computer vision. Regression and classification problems fall into the first and second categories, respectively, in supervised machine learning. However, the classic classification problem also applies to image classification. If necessary, localization follows the classification of an object in a photograph. Deep neural networks are used to identify a rectangular box around an item known as a bounding box. Feature extraction, classification, and localization are typically the next steps in this object detection problem. The object detector is known as a two-stage object detector. The first stage employs

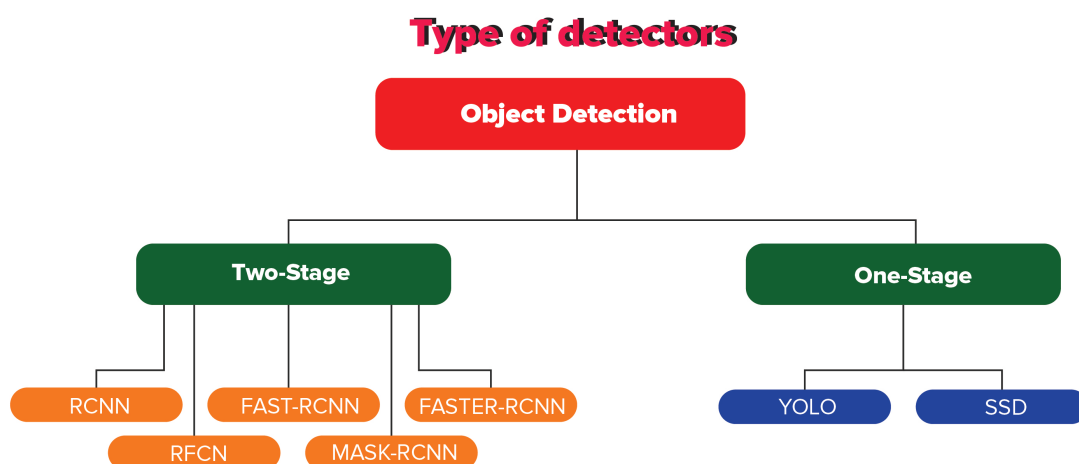


Figure 4.1: Classes of Detector

the region proposal network (RPN) to generate regions of interest (RoI), and the second stage is in charge of forecasting the items and defining the bounding boxes

for the scenario analysis. The purpose of the first stage is to choose believable region proposals by using different strategies, such as negative proposal sampling. One of the most popular models in this field is region-based convolutional neural networks (RCNN), which also includes Faster RCNN. Single-stage object detectors benefit from a simpler architecture designed specifically for single-stage object identification, which takes into account all region suggestions. These detectors generate the bounding boxes and class-specific probability for the fundamental objects by taking into account all of the spatial sizes of an image at once. Two-stage object detectors, on the other hand, outperform single-stage object detectors because they only focus on highly probable locations for object identification. However, efforts to complete this task in a single step or shot have received high praise since the development of You Only Look Once (YOLO) and its offspring. With the help of deep neural networks, these techniques present the localization problem as a regression problem. Other algorithms, in addition to YOLO, use Single Shot Detector (SSD) to identify objects.

4.2 YOLO

Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi published the acclaimed research paper[2] "You Only Look Once: Unified, Real-Time Object Identification" in 2015 that popularized the cutting-edge real-time object detection technique known as You Only Look Once (YOLO). YOLO (You Only Look Once) was created by Joseph Redmon using a unique architecture known as Darknet. Darknet, a highly adaptable research framework built in low-level programming languages such as YOLO, YOLOv2, YOLOv3, YOLOv4, YOLOv5, and, most recently, YOLOv7, has produced the best real-time object detectors in computer vision. The

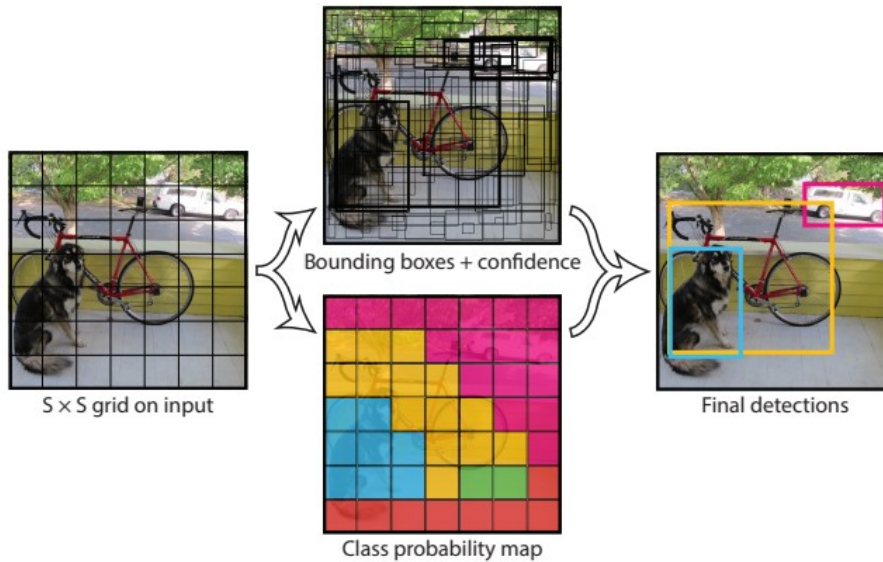


Figure 4.2: Hob, Dividing $S \times S$ grid and Predict object

YOLO series of models has developed since its introduction in 2016. Notably, both YOLOv2 and YOLOv3 were written by Joseph Redmon. YOLO models following

YOLOv3 are developed by new authors and have varied goals based on the authors that published them, as opposed to having strictly sequential releases to YOLOv3. The $S \times S$ area in each of the N grids created by the YOLO algorithm has the same size. These N grids are in charge of finding and locating the object they each contain. As a result, these grids can predict the name of the object, its likelihood of being in the cell, and its B bounding box coordinates in relation to its cell coordinates. Because it uses cells from the image for both detection and identification, this method significantly reduces computation. However, because many cells forecast the same item with different bounding box predictions, many duplicate predictions are produced. In order to solve this problem, YOLO uses non-maximal suppression. During non-maximal suppression, YOLO suppresses all bounding boxes with probability values below a specific cutoff. In non-maximal suppression, YOLO suppresses all bounding boxes with lower probability values. By weighing the likelihood ratings attached to each choice and choosing the highest one, YOLO achieves this. YOLO

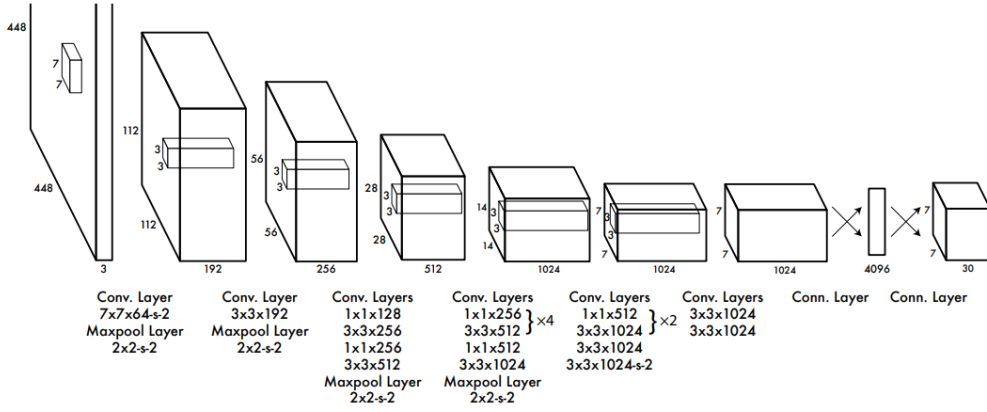


Figure 4.3: Network Architecture

accomplishes this by weighing the probability ratings associated with each option and selecting the highest one. The bounding boxes with the greatest Intersection over Union with the current high probability bounding box are then suppressed. Repeating this phase results in the acquisition of the last bounding boxes. The YOLO architecture, which is based on the GoogleNet design, has 24 total convolutional layers, with convolutional and pooling layers at the exact end. Instead of the GoogleNet's inception modules, YOLO only utilizes 11 reduction layers followed by 33 convolutional layers. The loss function is an integral aspect of the procedure. The YOLO method maximizes the loss function in Eq. 4.1 [4] during training.

$$\begin{aligned}
& \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^B 1_{ij}^{obj} [(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2] + \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^B 1_{ij}^{obj} [(\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2] \\
& + \sum_{i=0}^{S^2} \sum_{j=0}^B 1_{ij}^{obj} [(C_i - \hat{C}_i)^2] \\
& + \lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^B 1_{ij}^{noobj} [(C_i - \hat{C}_i)^2]
\end{aligned}$$

$$+\sum_{i=0}^{S^2} 1_i^{obj} \sum_{c \in classes} [(p_i(c) - \hat{p}_i(c))^2] (4.1)$$

To provide a brief explanation of the loss function, we break it into three sections. Equation (4.2) illustrates the major component.

$$\lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^B 1_{ij}^{obj} [(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2] + \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^B 1_{ij}^{obj} [(\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2] \quad (4.2)$$

It reflects the disappearance of the bounding box. It assists the network in learning the weights required to estimate the bounding box's coordinates and size. The next part is in Eq. (4.3):

$$\sum_{i=0}^{S^2} \sum_{j=0}^B 1_{ij}^{obj} [(C_i - \hat{C}_i)^2] + \lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^B 1_{ij}^{noobj} [(C_i - \hat{C}_i)^2] \quad (4.3)$$

It stands for the loss of object confidence. This is in charge of training the network to learn the weights and forecast if an item is present in a bounding box or not. The last part of the equation is given in Eq. (4.4):

$$\sum_{i=0}^{S^2} 1_i^{obj} \sum_{c \in classes} [(p_i(c) - \hat{p}_i(c))^2] \quad (4.4)$$

This is an example of categorization loss. The network will learn to predict the proper class for each of the bounding boxes as a result.

4.3 Pytorch

For our research we used YOLOv7. YOLOv7 is implemented using the PyTorch library. PyTorch is a popular deep learning framework that allows for easy implementation of complex neural networks[7]. The YOLOv7 network architecture is implemented as a PyTorch module, which is a class that inherits from the PyTorch nn.Module class. This allows the user to easily define the architecture of the network and use PyTorch's built-in functions for forward propagation and backpropagation. To use YOLOv7 in PyTorch, the user can define the architecture of the network using PyTorch's nn.Module class and then train the model using PyTorch's built-in optimizer and loss functions. PyTorch's automatic differentiation feature also allows for easy implementation of backpropagation, which is the algorithm used to train neural networks. Pytorch's support for CUDA also allows for easy use of GPU acceleration when training YOLOv7 which makes the training process faster[12]. PyTorch also provides a number of other features such as support for distributed training, which allows for parallel training on multiple GPUs or machines, and support for data loading and processing, which makes it easy to work with large datasets. PyTorch allows for easy implementation of complex neural networks, such as YOLOv7, by providing a number of built-in functions and classes for defining the architecture of the network, training the model, and performing other tasks. PyTorch's automatic differentiation feature and support for CUDA also make it easy to train YOLOv7 on large datasets and on multiple GPUs or machines. PyTorch also provides a number

of other features such as:

1. A built-in support for CUDA, which allows for GPU acceleration of computations
2. A built-in support for distributed training, which allows for parallel training on multiple GPUs or machines
3. A built-in support for data loading and processing, which makes it easy to work with large datasets
4. A built-in support for automatic differentiation, which allows for easy implementation of backpropagation, which is the algorithm used to train neural networks

PyTorch is also popular because it allows for easy interoperability with other Python libraries and frameworks, such as NumPy and scikit-learn, which makes it easy to use in a variety of different applications and environments.

Chapter 5

Methodology

5.1 Dataset

Every AI model must have data, which is also the main driver of machine learning's current popularity boom. Due to the availability of data, scalable ML algorithms are now workable as full - fledged solutions that might enhance a company's operations rather than being a by-product of those operations. As there weren't enough sources to get the defective clothing photographs we needed, we had some difficulties gathering our data. However, we were able to collect around 7115 dataset images for the model training. It is important to note that since we are dealing with three types of defects seams, holes, and stitches all 6600 data sets have the sort of photos where some garments have defects like holes and seams, while others have issues with stitching. Nearly 80% of the dataset was our own creation, while the remaining photos were gathered from various clothing and online platforms like TILDA and KAGGLE. We are training our algorithm to recognize the defects in our cloth using these defect photographs.

5.1.1 Process of Data collection

We generated 80% of the data by our own.. First and foremost, we visited a few garments to collect data, but not enough data was obtained. After that, we, the five members of our group, began to collect the data on our own. First, we tried to find the defects in our own clothes. Holes were easily obtained using this technique, but it was difficult to obtain the image of the stitches, for which we used our old clothes and cut the threads of the dress. Finally, we tried to collect a few data from different sites to increase the amount of data. The majority of our data was collected in this manner in order to train our model.

5.1.2 Hassle of Data Collection

We previously stated that we had only collected 20% of the data from online websites. Most of the sites did not have a suitable data set for us. Some websites provide fabric data sets, but we must subscribe to those websites in order to access the data, and the data were not particularly useful to us. As previously stated, the images were not clear and were blurry, which will not provide good accuracy from the model or cause the model to fail to understand the image. We visited some of

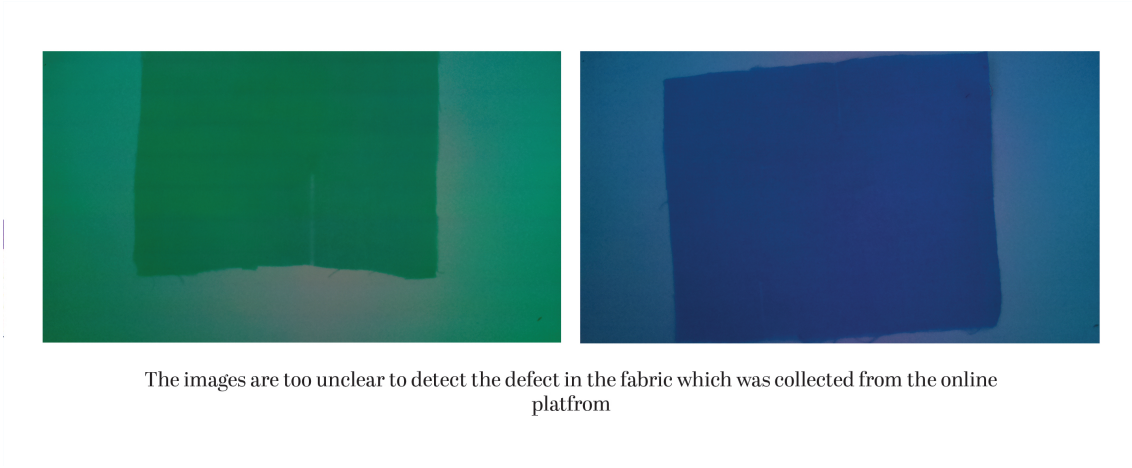


Figure 5.1: Unclear image collected from online

the garments to collect the default fabric dataset, but we did not get as much data as we expected because the supervisor stated that "they remove every day defect product in that day," for which we did not get many images, and it was very time consuming to visit the garments every day because most of the garments were in Gazipur, and it was costly to do this task in this way. Again, there is no data for the defect of holes, stitches, and seams created every day. As previously stated, we were working with three types of defects: holes, stitches, and seams. However, the most sensitive part was the holes and defects because the holes could be as big as a dot or as small as a dot, so to get a perfect image we needed a good quality camera as well as an environment, which we did not have because the camera could not focus on the tiny tough, resulting in a blurry image of that defect part. It was most crucial to keep the colour balance of the clothes and as well as the stitches because higher accuracy could be obtained if some fixed colour combination of clothes and stitches were used. However, due to the camera's low configuration and the environment, we were unable to obtain an accurate colour of both the stitches and the fabric. For example, we attempted to photograph a black cloth that was stitched with red ribbon, but after taking the photograph, we discovered that the ribbon colour was Magenda and the cloth colour was Midnight Blue.

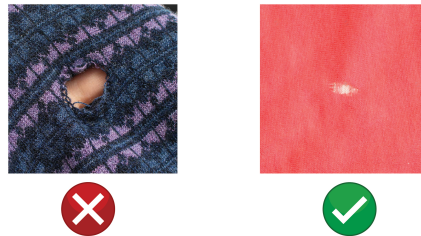


Figure 5.2: Pattern cloth image vs plain cloth image

Figure 5.3: Images and annotations

One of the biggest problems with data collection is that we must delete images of printed-type textiles from the collection since the model might not recognize the error in the bounding box owing to the print style of the fabric.

5.1.3 Data annotation

The practice of annotating data in machine learning to demonstrate the outcome we want our machine learning model to predict is known as data annotation. We are labeling, tagging, transcribing, or processing a dataset with the attributes we want our machine learning system to understand and recognize. When our model is put to use, we want it to recognize those features and decide what to do next. Annotated data displays characteristics that our algorithms will use to identify similar characteristics in unannotated data. Data annotation is used in supervised learning models and hybrid, or semi-supervised, machine learning architectures that include supervised learning. To annotate the data we have used Roboflow. Roboflow is a cloud-based workflow management system that coordinates the robots' AI-enhanced development pipelines. RoboFlow is data-centric, in contrast to the majority of conventional robotic development techniques, which are largely process-centric. This remarkable trait makes it particularly ideal for creating data-driven AI-enhanced robots[18]. The process of going from raw photos to a trained and deployed computer vision model is substantially sped up by the self-serve annotation tool called RoboFlow, which is provided with all Roboflow accounts. We no longer need to download a different software or export and re-import our photos in order to label a complete dataset or fix a single annotation. We can now do it within Roboflow. We have labeled or annotated our data using the bounding boxes, which is to say, we have indicated the image's faulty region using the bounding box to help the model identify the problem and the target object, and IOU Calculation. In the diagram

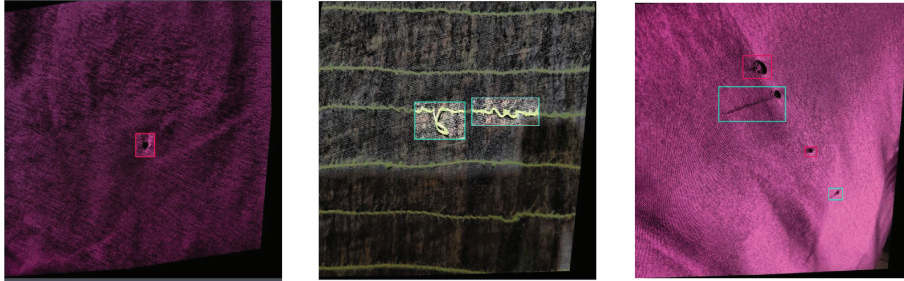


Figure 5.4: Bounding box through Roboflow

below we can see that we have a total of 7715 images and the annotations number is 9516 which means in the images we have 9516 defects and on average around for each image there is 1.3 annotations among the 3 classes that exist and among this 7115 images, there is no image where we can find any no defects which means all our images are of defect images and for which all images are annotated which are 0 missing annotations.

In the heat map of the annotation, we can see that almost all of our data which means the defects appeared at the center of our images. Through this, we can easily know that if our dataset images were resized or cropped then there would be no risk of data loss.

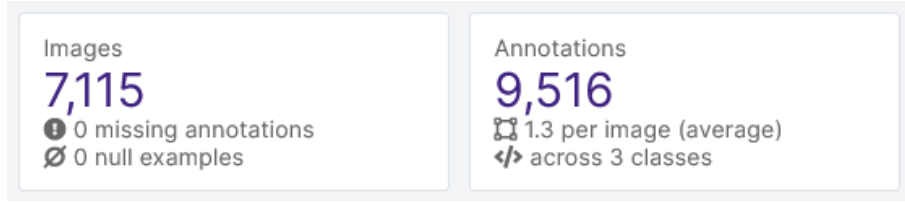


Figure 5.5: Number of images and annotations

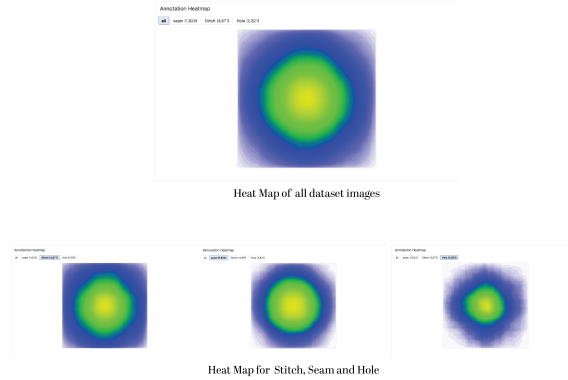


Figure 5.6: Heat Map Image for Hole, Seam and Stitch

5.1.4 Labeled Classes

By striking a balance between the common classes, we annotated 3 distinct object classes. In order to decrease the number of empty classes and ensure that we covered the faulty portions of each picture, we also tried to limit the number of annotation attempts. We did, however, meticulously mark each occurrence with occlusion. Here



Figure 5.7: Number of data in each class

in diagram 5.7 we see that among the 7115 images 4871 data are of stitches which is a pretty good number for training a model as well as for hole also which is 2821 in number but for the dataset seam the number of data is a bit low.

Data pre-processing and augmentation

Images must be preprocessed before they can be used for model training and inference. Preprocessing is an important step in preparing image data for use in a computer vision model. Preprocessing is required for technical as well as performance reasons. All images must be the same size arrays in order for convolutional neural networks, a commonly used software vision design, to have fully linked layers. If our photographs are not all the same size, the model may not work properly. If the image is not the same size as the model we are building in code with a library like Tensorflow, we will most likely encounter problems. Therefore, we stretched the image to 640x640 format in order to resize it and enhance the data as images

should be compressed for reduced file sizes and quicker training. To improve our data, the technique Auto adjust contrast was used because when utilizing contrast preprocessing, edges become more distinct as the differences between nearby pixels are emphasized. In general, flattening the contrast of the picture using preprocessing is beneficial if an issue is thought to contain low-contrast photos or a section of images with saturated contrast. Again we have used Modify classes to Remap class labels and remove them, and lastly, filter null was used as we need some blank photos. Not much, which allows our model to understand that an item isn't constantly there. We also used Auto-Orient to standardize pixel ordering and discard EXIF rotations. Here the average size of each image is 0.41 megapixels whereas the highest and lowest both are 0.41-megapixel images. In the above images, we can see that all of the dataset images are of square shapes and its dimension is 640px by 640px. In order to expose the model to a wider range of training, pic-

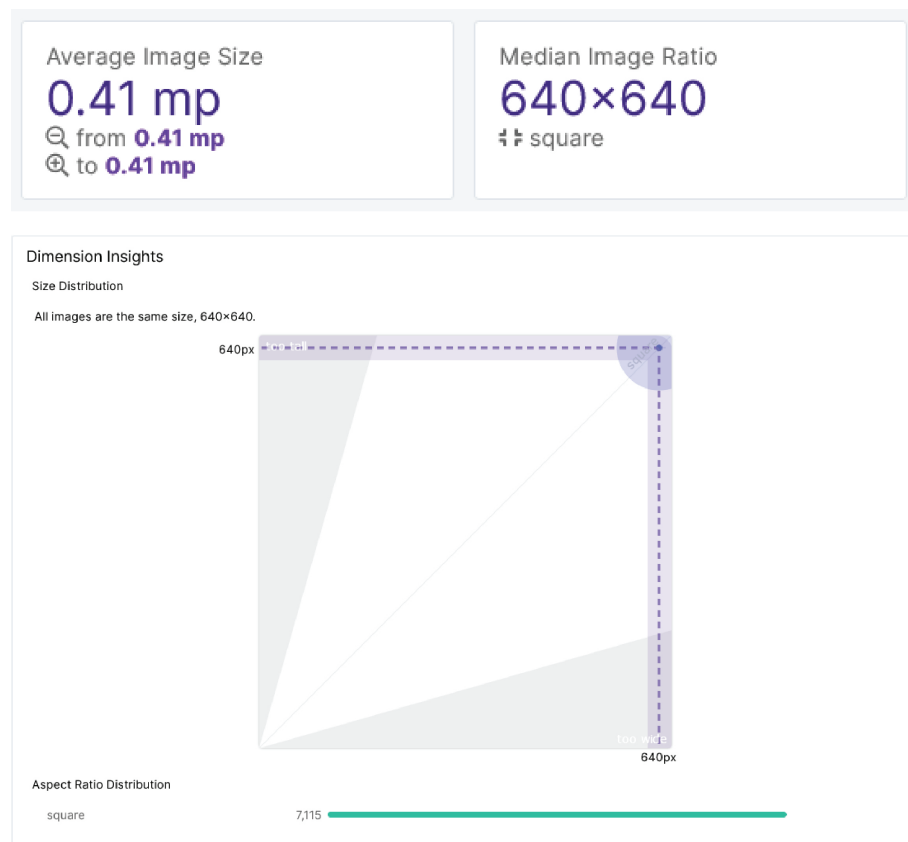


Figure 5.8: Size and quality of the images

ture augmentation is the use of changes to photos to produce different versions of the same information. It is challenging to completely capture a picture of a model that includes every conceivable real-world event. This is where augmentation can be useful. By improving our photographs, we can increase the sample size of our training data and include additional situations that may be difficult to uncover in the real world. By expanding the training data, the model can learn from a broader range of events and generalize to new situations. In the augmentation, we flipped our data into a horizontal format because flipping a picture (and its annotations) is a fairly simple technique that can greatly improve model performance. Our models are discovering which clusters of pixels and the connections among those clusters

indicate the presence of an item in the picture. However, machine learning models, such as convolutional neural networks, have a propensity to be quite fragile. For example, although our models may memorize a particular arrangement of pixels that describes an object, if that same object is mirrored throughout the image, our models may find it difficult to identify it. By producing several copies of our photos in various orientations without having to go through the time-consuming process of obtaining and labeling new training data, we give our deep learning model more data to learn from. Turn both clockwise and counterclockwise to a 90° angle as Additionally, random rotation may assist prevent any overfitting. Random rotation may enhance variance to stop a model from forgetting your training data, even while the camera location is constant in relation to the subjects your model is experiencing. The data augmentation technique known as random crop was used to construct a random subset of the original picture, cropping it to 0% at the lowest zoom and 20% at the highest magnification. This helps our model generalize since the object(s) of interest that we want our models to learn aren't always entirely visible in the picture or the same scale in our training data. rotated it 15 degrees to 15 degrees because Even when the camera and the topic are not exactly in focus, it aids your model in detecting things. and then sheared it to 15 degrees horizontally and 15 degrees vertically which To make our model be more resistant to camera and subject pitch and yaw, add variety to viewpoint.

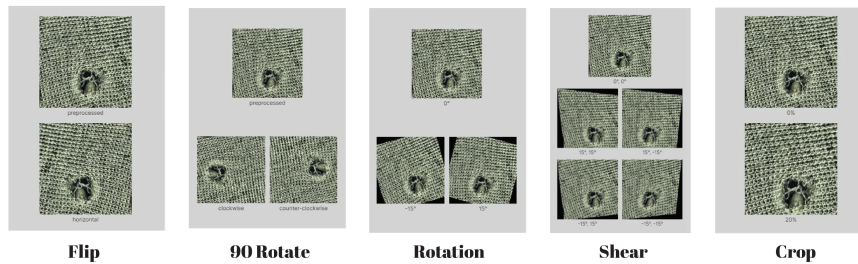


Figure 5.9: Applied augmentation techniques

5.1.5 Train-Valid-Test Split

The data will be utilized for training 70% of the time, validation 20% of the time, and testing 10% of the time. Using the validation set, we tweaked the hyper-parameters of our models to improve performance. The test set was used to determine the final performance of the models.

5.1.6 Data analysis

Here if we go through the histograms we can see that the Y axis represents the data that the number of the images and the x axis represent that number of defects in that images. Among all the images 5894 images contain a single defect data where rest of the of the data contain 2 to 5 defects where few contains 6-9 defects. In the case for the seam data-set almost 1611 data contain single defect of seam class and a very few contain 2 defect of seam class. In the case of stitch data-set we can see that almost 2994 images contain a single stitch class defect data where a very few images contain 2-4 defects for the stitches class and one or two images contain 5-7

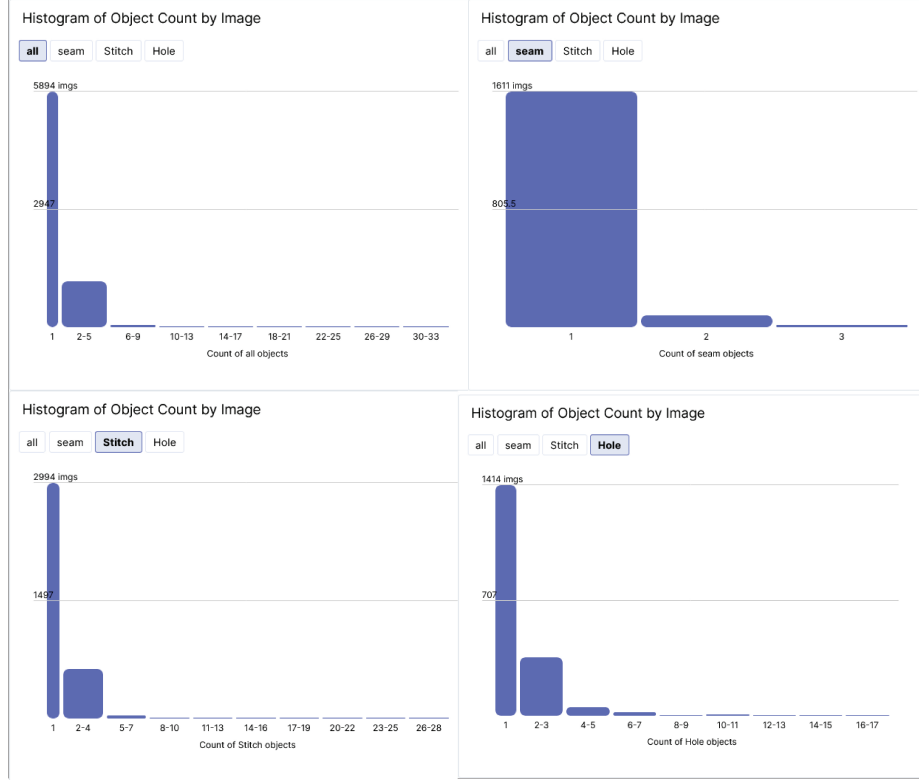


Figure 5.10: Graph for data analysis

defects of this class. Lastly For the Hole data-set 1414 images contain a single defect where around 300 to 350 images contain 2 to 3 defects of the Hole class and a few contain 4 to 5 and 6 to 7 defects of this Hole class. In the end it can be said that almost all images contain a single defect among these 3 classes.

5.1.7 YOLOv7

As previously stated, several successors to YOLO have emerged since its initial release. In terms of accuracy and speed, YOLOv7 outperforms all other object detection algorithms on the benchmark. In July 2022, Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao [20] published the official YOLOv7 article, "YOLOv7: Trainable bag-of-freebies establishes new state-of-the-art for real-time object detectors." YOLOv7 significantly improves real-time object recognition accuracy while lowering inference costs. As previously demonstrated in the benchmarks, YOLOv7 successfully outperforms other well-known object detectors by reducing the number of parameters and processing required for cutting-edge real-time object detections by approximately 40% and 50%, respectively. This allows it to reach conclusions more quickly and with greater detection precision. In general, YOLOv7 offers a quicker and more dependable network architecture that offers an improved feature integration method, more accurate object identification performance, a more dependable loss function, and improved label assignment and model training efficiency. In terms of computing hardware, this makes YOLOv7 far more affordable than other deep learning models. It can be trained on small datasets much more rapidly without any pre-learned weights. Bag-of-freebies features, which boost accuracy without slowing down detection speed, were included in YOLOv7. Because

of this, YOLOv7 is faster and more accurate than earlier YOLO versions.

5.2 YOLOv7 Architecture

Prior YOLO model designs, such as YOLOv4, Scaled YOLOv4, and YOLO-R, were used to develop the YOLOv7 architecture. The Extended Efficient Layer Aggregation Network (EELAN) is the computational component of the YOLOv7 backbone (E-ELAN). The YOLOv7 E-ELAN architecture supports the model’s learning ability by utilizing ”expand, shuffle, merge cardinality” to achieve the capacity to continuously boost the learning ability of the network without altering the initial gradient path. The basic goal of model scaling is to change important model char-

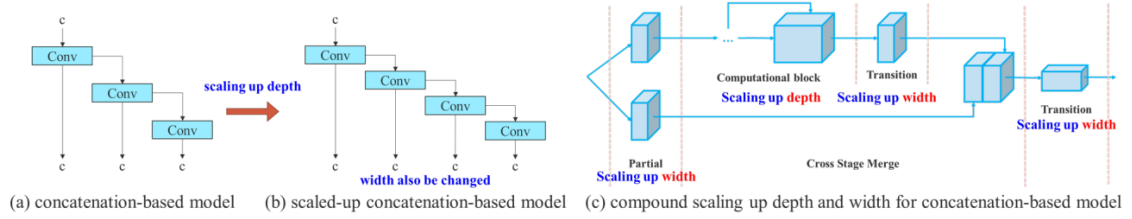


Figure 5.11: YOLOv7 Compound Model Scaling

acteristics in order to create models that are appropriate for a variety of application requirements. Model scaling can improve the model’s resolution, depth, and breadth (number of stages, for example) (input image size). Conventional approaches based on concatenation-based architectures (such as ResNet or PlainNet) must consider various scalability factors concurrently rather than individually. For example, increasing model depth alters the ratio of input to output channels in a transition layer, resulting in the model using less hardware. For a concatenation-based model, YOLOv7 includes compound model scaling. The compound scaling strategy can keep the model’s optimal structure while preserving its original design characteristics. Scaling a compound model is accomplished as follows: Changing the depth factor of a computational block, for example, necessitates modifying the block’s output channel. The width factor scaling is then applied with the same level of modification to the transition layers.

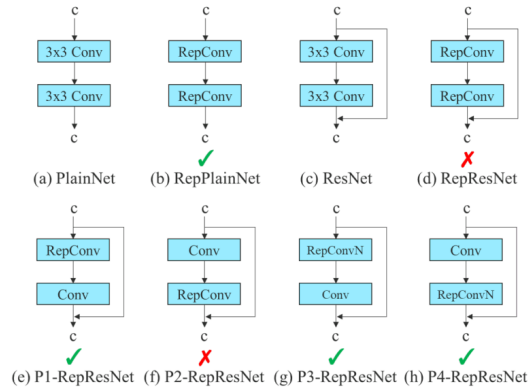


Figure 5.12: RepConv

RepConv is a planned re-parameterized convolution method that does not require identity connection (RepConvN). The goal is to avoid identity connections in re-parameterized convolution when residual or concatenation are used to replace a convolutional layer.

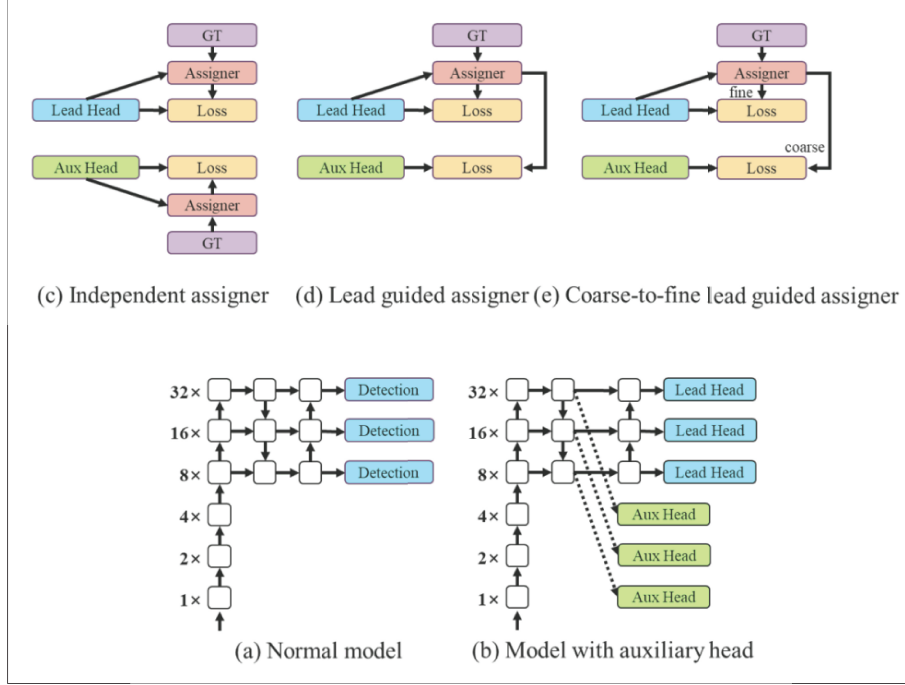


Figure 5.13: RepConvLead Head and Auxiliary Head

A head, a neck, and a backbone are the three main structural parts of a YOLOv7. The head is where the anticipated model findings are. YOLOv7 is not constrained to a single head since it was inspired by Deep Supervision, a method often used in training deep neural networks. The lead head produces the end product, while the auxiliary head aids in middle-layer training. The Label Assigner technique, which delivers soft labels after taking into consideration ground truth and network prediction outcomes, was created to enhance deep network training. Reliable soft labels employ calculation and optimization strategies that take into account both the ground truth and the quality and distribution of prediction output. Conventional label assignment, on the other hand, makes direct use of the ground truth to produce hard labels based on pre-established standards.

As a starting point, the performance of YOLOv7 was compared to that of YOLOv4, YOLOv5, and YOLOR. The same training processes were used to all of the models. The new YOLOv7 boasts the greatest speed-to-accuracy balance when compared to cutting-edge object detectors. YOLOv7 beats all prior object detectors in terms of speed and accuracy with frame rates ranging from 5 to 160 frames per second. The YOLOv7 method offers the best accuracy of any real-time object recognition model currently in use and can operate at 30 frames per second or more when utilizing a GPU V100.

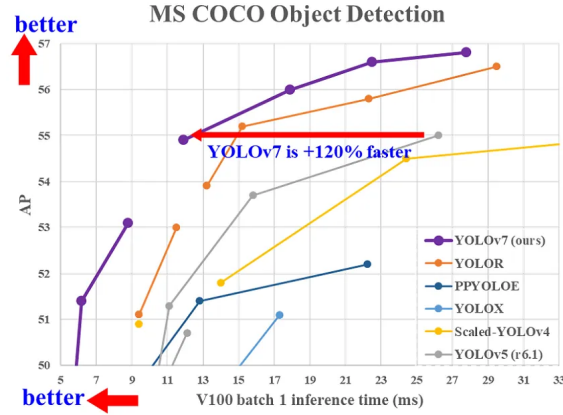


Figure 5.14: Comparison of Model and it's performance[20]

5.3 Model Description

The input image is first resized to the 640*640 input size of the Backbone network. The Head layer will then generate three feature maps of varying sizes based on the information from Backbone. Finally, using REP and Conv, the prediction outcomes can be obtained. The model is improved by YOLOv7’s structural re-parameterization and dynamic label assignment. By examining the gradient flow propagation channels and the re-parameterization strategies of the various network modules, re-parameterized convolution is integrated with various modules. Because the model generates a large number of outputs for dynamic label assignment, assigning better dynamic targets to specific branches during training is difficult. To address this issue, the YOLOv7 developers proposed a novel dynamic label assignment technique known as coarse-to-fine lead-guided label assignment. A model

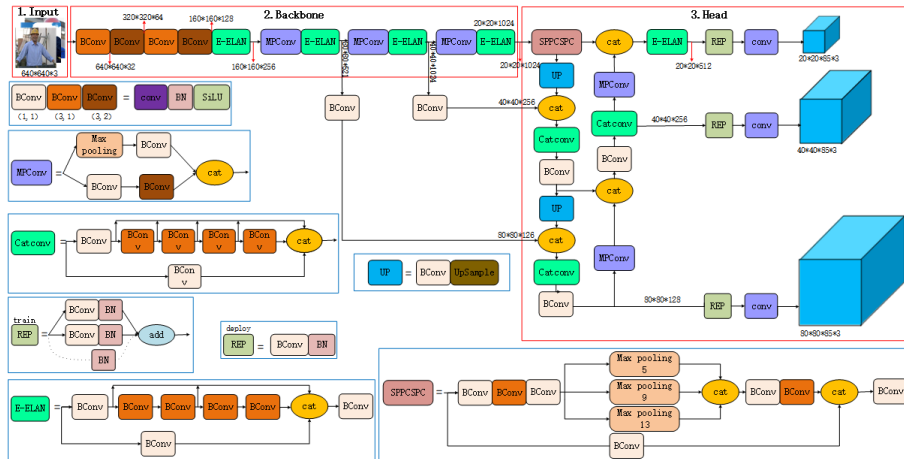


Figure 5.15: YOLOv7 network structure[20]

scaling for concatenation-based models is also recommended in addition to these two characteristics. By making better use of FLOPS and parameters, this approach significantly improves the network’s inference speed and detection accuracy. The model’s size was determined for the object identification job using the parameters in the weight file, and the model’s real-time performance was evaluated using FPS. the mAP was used in this study to assess detection precision.IoU was set to 0.5.

so mAP@0.5 was used to calculate the average AP value across all categories by calculating the AP of all photographs in each category. If the average mAP's IOU threshold is between 0.5 and 0.95, it is referred to as mAP@0.5:0.95. To estimate the number of model parameters, we examined the size of the pt weight file. The effectiveness of the model in identifying or evaluating model performance was evaluated in this article using Precision, Recall, and Accuracy. The following equations (1), (2), and (3) show their formulas:

$$Precision = \frac{TP}{TP + FP} \times 100\%.....(1)$$

$$Recall = \frac{TP}{TP + FN} \times 100\%.....(2)$$

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \times 100\%.....(3)$$

False Positive is abbreviated as FP, True Negative is abbreviated as TN, True Positive is abbreviated as TP, and False Negative is abbreviated as FN. The term "mAP" refers to the overall average of all AP values (mean Average Precision). To determine AP, our classification employs an interpolation accuracy of 11 points. The formula is shown in (4).

$$AP = \frac{1}{11} \sum_{r \in \{0,0.1,...,1\}} Pinter(r).....(4)$$

$$mAP = \frac{\sum_{q=1}^Q AP(q)}{Q}.....(5)$$

$$FPS = \frac{1000}{T(I) + T(N)}.....(6)$$

In this case, T(I) stands for the inference time, and T(N) for the NMS time.

$$F1 = \frac{2PR}{P + R} = \frac{2TP}{2TP + FN + FP}.....(7)$$

A large number of defect photos with labels including information about the defect's location and category are prepared prior to training the model. The data set is then divided into a training set for modifying the model's trainable parameters and a test set for assessing the model's training impact. The model's trainable parameters are first initialized at random. The training set is split up into a number of tiny batches of samples that each include n flawed photos in the second stage before being fed into the YOLOV7 model. As a result, one operation becomes one iteration, and the split training set is all used for training in one training round. The output results are then calculated with the labels to calculate the error through

the loss function and adjust the trainable model parameters using the back propagation method. The model's trainable parameters are fixed and tested using a test set after each training cycle to ascertain the model's effectiveness in the third stage. The technique is then ended when steps 2 and 3 are executed a second time and the model's detection performance stops improving. Using our primary dataset, we trained the YOLOv7 model. Backpropagation was performed with a batch size of 16, eight workers, an SGD optimizer, momentum set to 0, 0,937, and learning rates of 0.01 and 0.001 after hyperparameter modification. Following 4,5 hours of training on an RTX3080ti device and 200 iterations, the learning rate gradually dropped below 0.001. rates of 0.01 and 0.001 following hyperparameter adjustment. The learning rate eventually fell below 0.001 after 200 iterations and 4.5 hours of training on an RTX3080ti device.

Chapter 6

Model Training

We used two different pre-trained weights of YOLOv7 to evaluate our model and compared them in terms of accuracy and detection speed. We used YOLOv7 and YOLOv7x where YOLOv7 is for usual devices with a good GPU, and this one is the basic model, while YOLOv7x is for edge devices which consume less computation power than YOLOv7. This paper shows the comparison between these two models as our motive is to implement them in real-world operations, and this comparison will help to sort out accuracy and speed in terms of computation power and to design the implementation model.

Model Training environment	
Items	Parameter
OS	Windows 10 Professional
GPU	NVIDIA Ge Force RTX 3080ti
CPU	AMD Ryzen 9 5950X 16-Core
RAM	64GB
Programming Language	Python 3.9
Machine Learning Version	Pytorch 1.13.0 +CUDA 11.7

Figure 6.1: Information of environment

6.1 Yolov7

For this model, we first used YOLOv7.yaml from cfg and reduced the number of class parameters to 3 as initially, our dataset has 3 classes[16]. We used YOLOv7.pt as a pre-trained weight. We trained the model with a learning rate of 0.01 and 0.001, SGD as an optimizer, with a batch size of 16. The model accepts all the images with a fixed size of 640*640

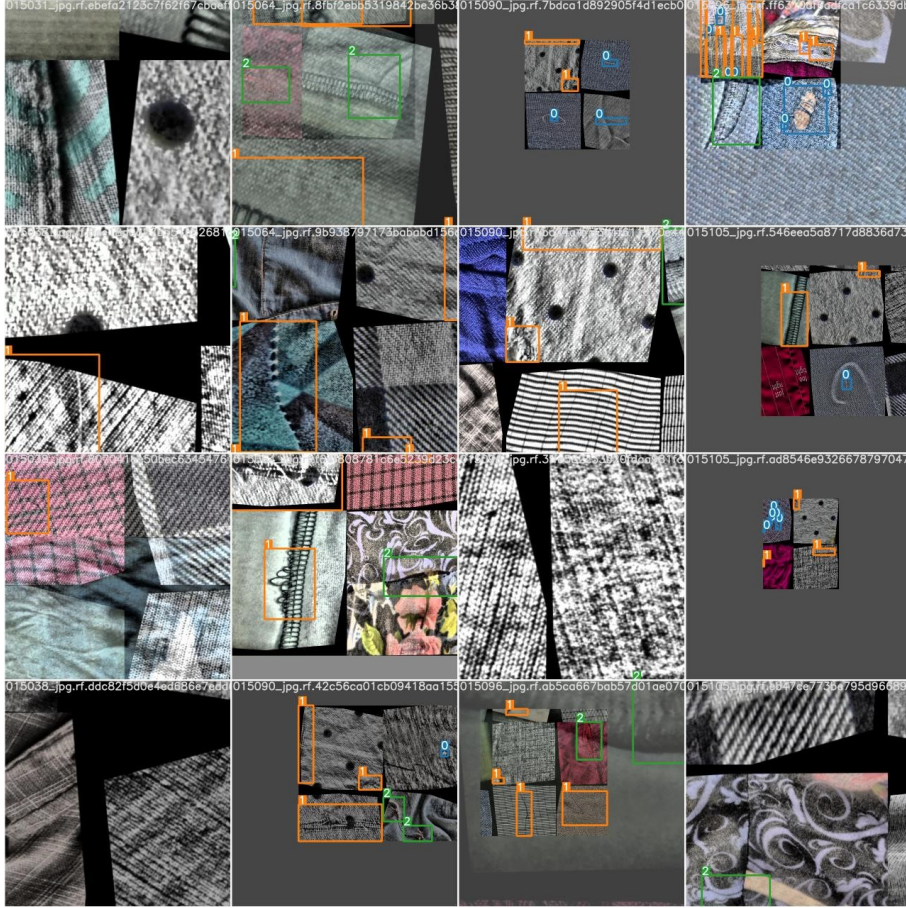


Figure 6.2: Training batch images for this YOLOv7

6.1.1 mAP@.5=0.39

For this the model was trained for 100 epochs. The model converged within 100 epochs, as shown in Fig. 6.3, which also displays precision, recall, and mean Average Precision (mAP) metrics across the training epochs on our dataset. A performance gauge for a machine learning model is precision is a metric used to assess how well a model predicts something favorable. Recall gauges how well a model can identify positive samples. Mean Average Precision is a widely used statistic for assessing object detectors' accuracy. In this context, "mAP at 0.5" refers to mAP calculated at the Intersection over the Union (IoU) threshold of 0.5, while "mAP at 0.5:0.95" refers to mAP derived by averaging different IoU values between 0.5 and 0.95 with a step of 0.05. Box loss, objectness loss, and classification loss are the three different forms of loss graphs. The bounding box coverage and accuracy of our model's ability to find the objects' centers are measured by the box loss. The likelihood that an item is present in a location suggested by the algorithm is essentially measured by objectness. Finally, classification loss gauges how well an algorithm correctly identifies an object's class. By the conclusion of 100 epochs, the training set's classification, objectness, and box losses had all decreased to very low levels. By the conclusion of 100 epochs, the box and classification loss of the validation set had fallen to a lower value, while the objectness loss of the validation set had continued to rise. After evaluating the validation set, YOLOv7 scored a precision of 0.484, recall of 0.391, mAP at 0.5 of 0.383, and mAP at 0.5:0.95 of 0.112 for all three classes. Fig

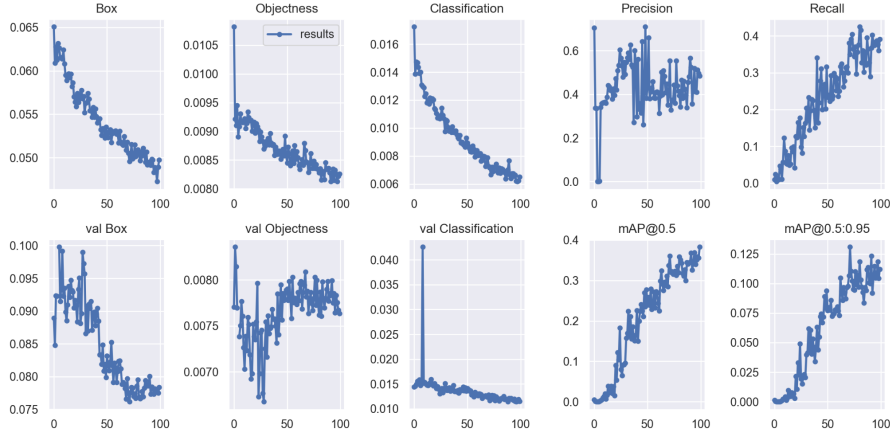


Figure 6.3: Graphic representation of initial experiment using YOLOv7

6.4 shows the distribution of the Average Precision (AP) at the IoU threshold of 0.5 among our three classes in the validation set, together with the related precision and recall values. The table shows that mAP at 0.5 for seam is 0.849 which is very high in comparison to the other two classes whereas the Hole class got 0.088 mAP at 0.5 which is too low with respect to the other two classes. And lastly Stitch class got 0.211 mAP at 0.5.

11.56	0.04976	0.008256	0.006513	0.06453	22	640: 100%
Class	Images	Labels	P	R	mAP@.5	mAP@.5:.95:
all	287	333	0.484	0.391	0.383	0.112
Hole	287	25	0.356	0.16	0.0888	0.0316
Stitch	287	207	0.374	0.222	0.211	0.056
seam	287	101	0.722	0.792	0.849	0.249

Figure 6.4: Results of YOLOv7 initial Experiment

Below fig 6.5 shows the Confusion matrix and 6.6 show the precision-recall graph, F1 graph, precision graph and recall graph respectively.

6.1.2 mAP@0.5=0.602

After evaluating the first training which mAP@0.5 was 0.39 we used the hyper-parameter tuning technique to increase our accuracy. For this, we changed the learning rate from 0.01 to 0.001. We also made changes in our dataset by re-annotating a large amount of data and also modifying the anchor box size to optimize varieties in its sizes. We also increased the data size of the Hole class and Stitch class and seam class as well for increasing the accuracy of these classes. For this, the model was trained for 200 epochs. The model converged within 200 epochs, as shown in Fig. 6.7, which also displays precision, recall, and mean Average Precision (mAP) metrics across the training epochs on our dataset. A performance gauge for a machine learning model is precision is a metric used to assess how well a model predicts something favorable. Recall gauges how well a model can identify positive samples. Mean Average Precision is a widely used statistic for assessing object detectors' accuracy. In this context, the terms "mAP at 0.5" and "mAP at 0.5:0.95" refer to the mAP computed at the Intersection over Union (IoU) threshold of 0.5 and "mAP

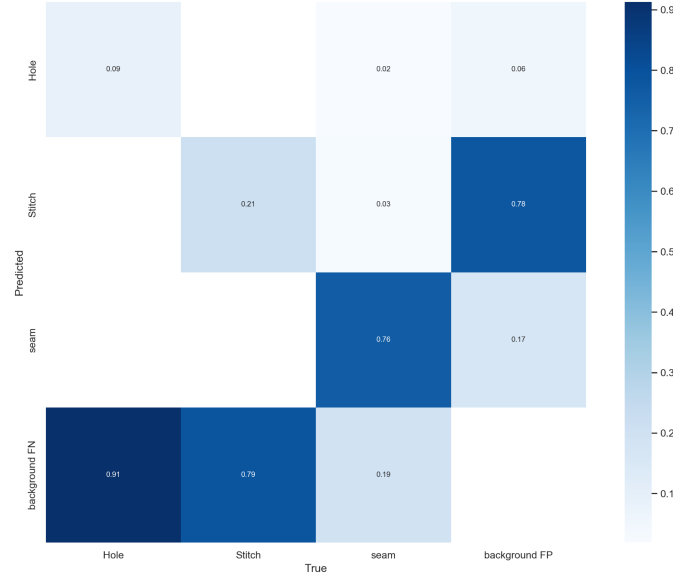


Figure 6.5: Confusion Matrix

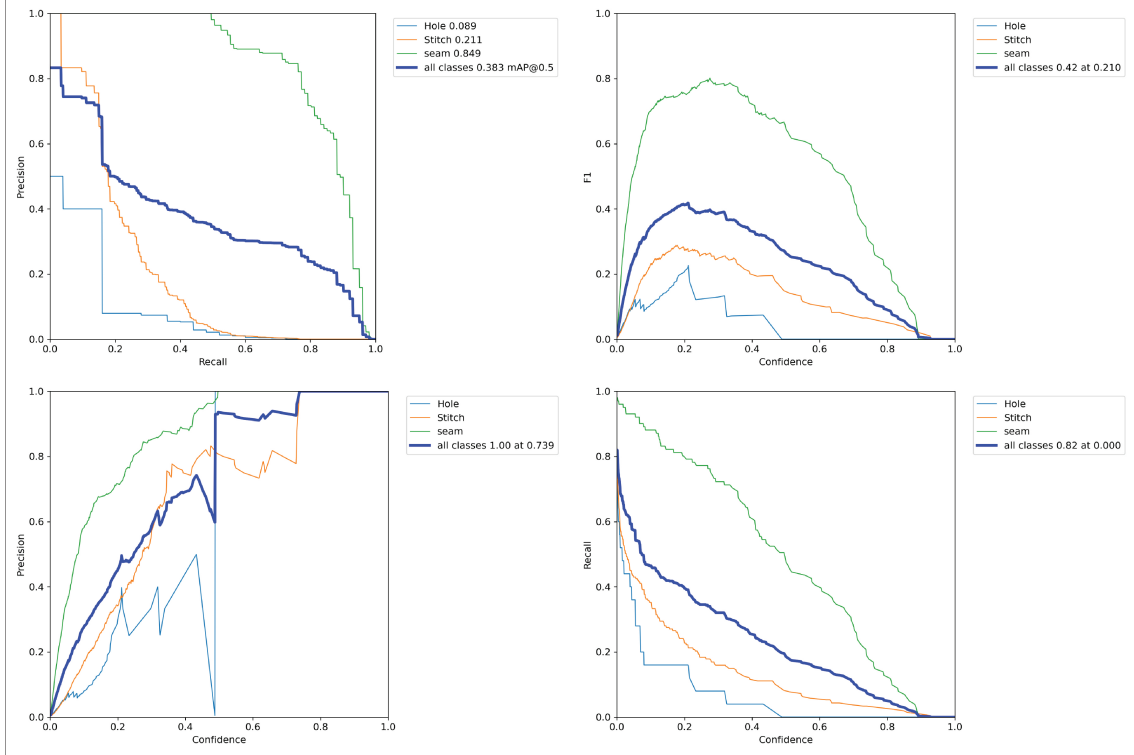


Figure 6.6: Precision-recall graph, F1 graph, precision graph and recall graph

at 0.5:0.95” refers to the mAP computed by averaging over different IoU values between 0.5 and 0.95 with a step of 0.05, respectively. Box loss, objectness loss, and classification loss are the three different forms of loss graphs. The bounding box coverage and accuracy of our model’s ability to find the objects’ centers are measured by the box loss. The likelihood that an item is present in a location suggested by the algorithm is essentially measured by objectness. Finally, classification loss gauges how well an algorithm correctly identifies an object’s class. By the conclusion of 200

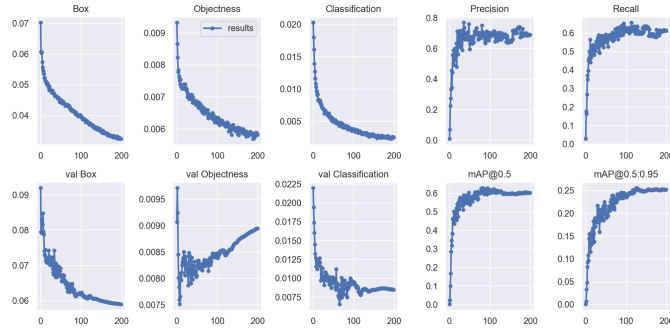


Figure 6.7: Results of YOLOv7 on validation set

epochs, the training set's classification, objectness, and box losses had all decreased to very low levels. By the conclusion of 200 epochs, the box and classification loss of the validation set had fallen to a lower value, while the objectness loss of the validation set had continued to rise. After evaluating the validation set, YOLOv7 scores precision of 0.69, recall of 0.612, mAP at 0.5 of 0.602, and mAP at 0.5:0.95 of 0.252 for all three classes. The distribution of Average Precision (AP) at the IoU threshold of 0.5 for our three classes in the validation set is shown in Fig. 6.8, along with the associated precision and recall values for each class. Fig. 6.8 shows that mAP at 0.5 for seam is 0.96 which is still very high in comparison to the other two classes. But the interesting part is after the approach of tuning parameters and changes in the dataset the mAP@0.5 of the Hole class increases significantly up to 0.353 from 0.088. mAP@0.5 of Stitch class also increases up to 0.493 from 0.211. This signifies that changes in the learning rate and dataset as well as increasing the number of epochs improved the accuracy of the model. Below fig 6.9 shows

Class	Images	Labels	P	R	mAP@.5	mAP@.5:.95
all	362	473	0.69	0.612	0.602	0.252
Hole	362	138	0.479	0.428	0.353	0.147
Stitch	362	233	0.642	0.484	0.493	0.193
seam	362	102	0.95	0.923	0.96	0.416

Figure 6.8: Results of YOLOv7 on validation set

Confusion matrix and 6.10 show precision-recall graph, F1 graph, precision graph and recall graph respectively.

6.2 YOLOv7x

YOLOv7 is for basic consumers of gpu. While for edge devices with less computational YOLOv7x was introduced in their paper of YOLOv7[20]. For this model, we first used YOLOv7s.yaml from cfg and reduced the number of class parameters to 3 as initially, our dataset has 3 classes. We used YOLOv7.pt as a pre-trained weight. We trained the model with a learning rate of 0.001, SGD as an optimizer, with a batch size of 8. The model accepts all the images with a fixed size of 640X640.

For this, the model was trained for 100 epochs. The model converged within 100

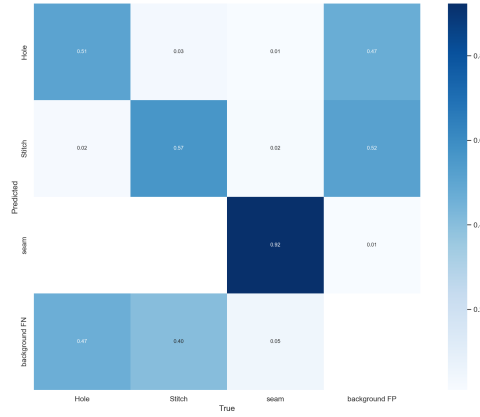


Figure 6.9: Confusion Matrix

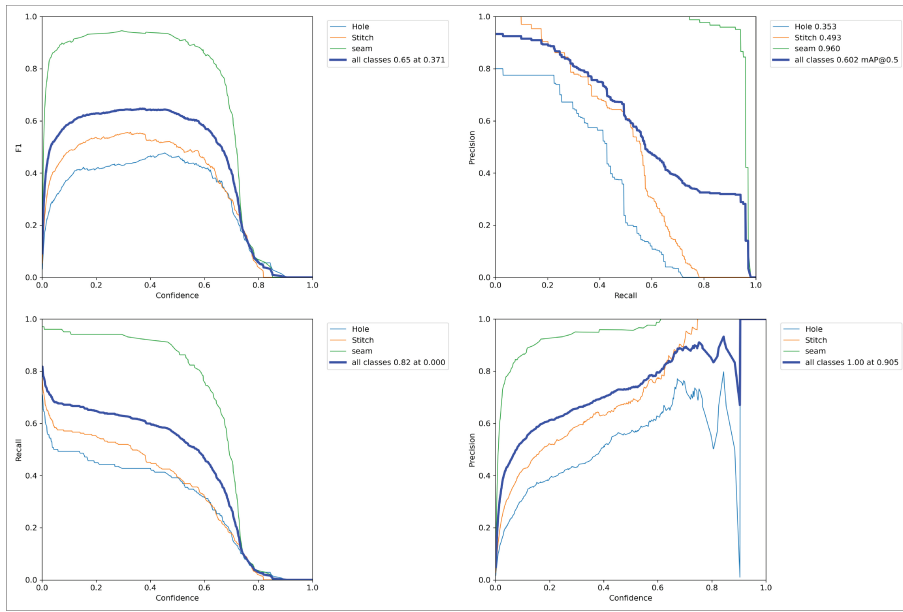


Figure 6.10: Precision-recall graph, F1 graph, precision graph and recall graph

epochs, as shown in Fig. 6.12, which also displays precision, recall, and mean Average Precision (mAP) metrics across the training epochs on our dataset. A performance gauge for a machine learning model is precision is a metric used to assess how well a model predicts something favorable. Recall gauges how well a model can identify positive samples. Mean Average Precision is a widely used statistic for assessing object detectors' accuracy. In this context, the terms "mAP at 0.5" and "mAP at 0.5:0.95" refer to the mAP computed at the Intersection over Union (IoU) threshold of 0.5 and "mAP at 0.5:0.95" refers to the mAP computed by averaging over different IoU values between 0.5 and 0.95 with a step of 0.05, respectively. Box loss, objectness loss, and classification loss are the three different forms of loss graphs. The bounding box coverage and accuracy of our model's ability to find the objects' centers are measured by the box loss. The likelihood that an item is present in a location suggested by the algorithm is essentially measured by objectness. Finally, classification loss gauges how well an algorithm correctly identifies an object's class. By the conclusion of 100 epochs, the training set's classification, objectness, and box losses had all decreased to very low levels. By the conclusion of 100 epochs,

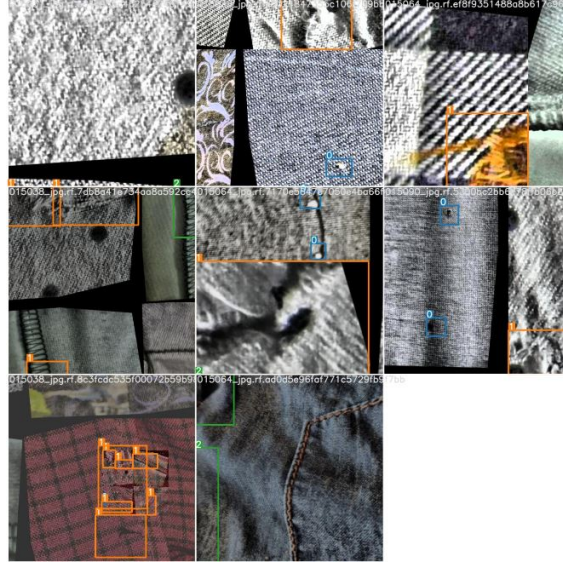


Figure 6.11: Training batch images for this YOLOv7x

the box and classification loss of the validation set had fallen to a lower value, while the objectness loss of the validation set had continued to rise. After evaluating in

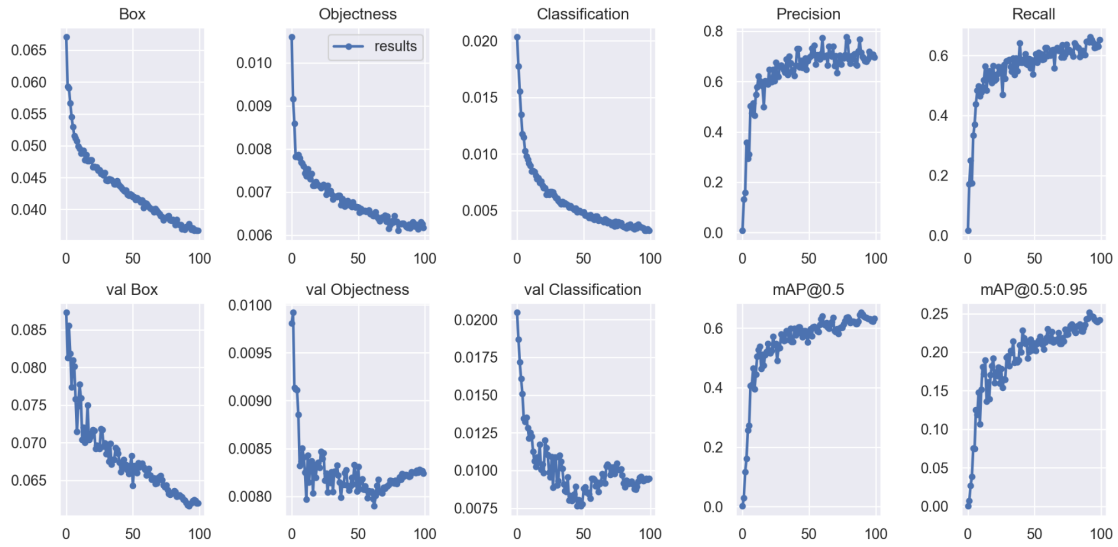


Figure 6.12: Graphic representation of experiment using YOLOv7x

the validation set, YOLOv7 scores precision of 0.697, recall of 0.652, mAP at 0.5 of 0.633, and mAP at 0.5:0.95 of 0.242 for all three classes. In the validation set, our three classes are broken down into Average Precision (AP) at an IoU threshold of 0.5, along with the matching precision and recall values. Fig. 6.13 shows that mAP at 0.5 for seam is 0.95 which is still very high in comparison of the other two classes. But the interesting part is after the approach of tuning parameters and making changes in the dataset the mAP@.5 of the Hole class increases significantly up to 0.44 while in YOLOv7 it was 0.353. mAP@.5 of Stitch class also increases up to 0.497. This signifies that YOLOv7x has better accuracy in comparison to YOLOv7 for our model.

Class	Images	Labels	P	R	mAP@.5	mAP@.5:.95:
all	362	473	0.697	0.652	0.633	0.242
Hole	362	138	0.522	0.536	0.444	0.149
Stitch	362	233	0.63	0.489	0.497	0.186
seam	362	102	0.938	0.931	0.957	0.391

Figure 6.13: Results of YOLOv7x Experiment

Below fig 6.14 shows the Confusion matrix and 6.15 show the precision-recall graph, F1 graph, precision graph and recall graph respectively.

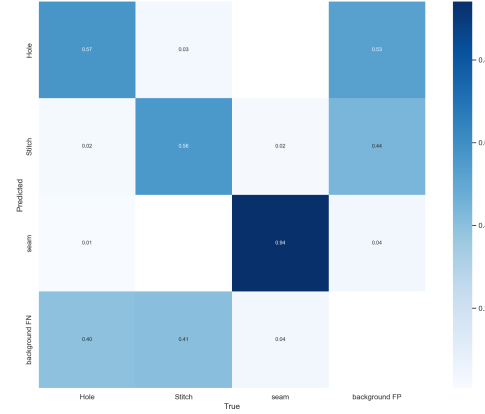


Figure 6.14: Confusion Matrix

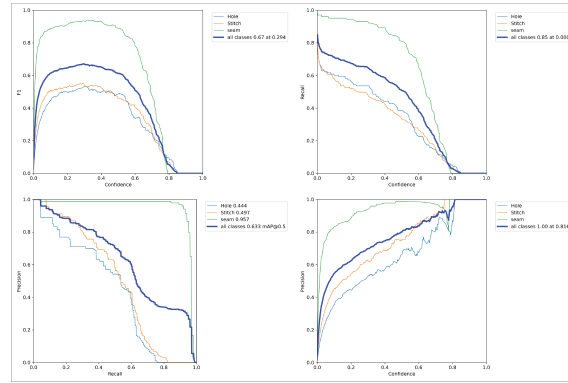


Figure 6.15: Precision-recall graph, F1 graph, precision graph and recall graph

6.3 YOLOv7-w6

For this model, we used YOLOv7-w6.yaml from cfg and YOLOv7-w6.pt as a pre-trained weights. We trained the model with train_aux.py and with a learning rate of 0.001, SGD as an optimizer, with a batch size of 8. The model accepts all the images with a fixed size of 640X640. For this model, we initially trained the model with an epoch of 100 but the validation AP was 31% and from the resulting graph we observed that the AP rate was increasing til the last epoch. So to increase the AP more we included more data in our dataset so that we can train our model with more epochs. Then the model converged within 250 epochs, as shown in Fig. 6.17,

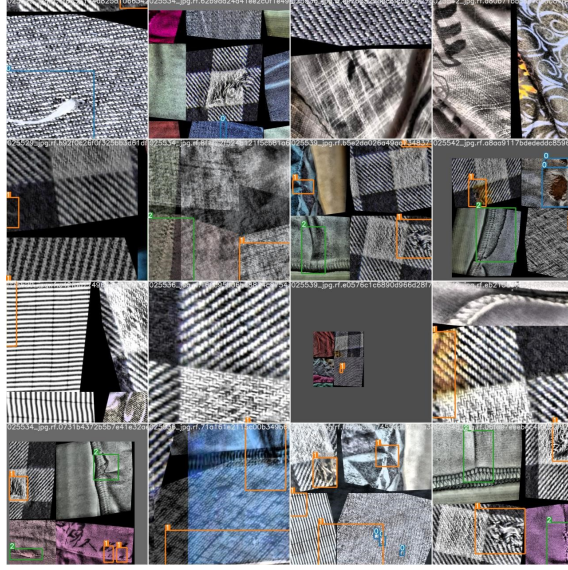


Figure 6.16: Training batch images for YOLOv7-w6

which also displays precision, recall, and mean Average Precision (mAP) metrics across the training epochs on our dataset. By the conclusion of 250 epochs, the box and classification loss of the validation set had fallen to a lower value, while the objectness loss of the validation set had continued to rise. After evaluation in fig.

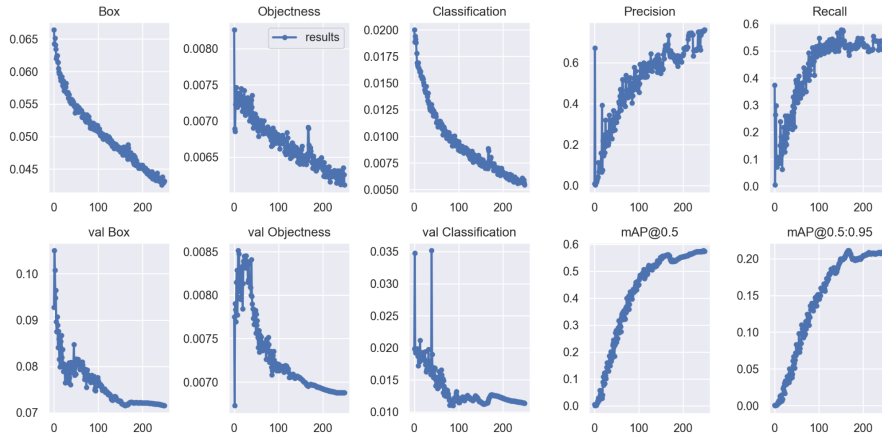


Figure 6.17: Graphic representation of experiment using YOLOv7-w6

6.18 the validation set, YOLOv7-w6 scores precision of 0.76, recall of 0.51, mAP at 0.5 of 0.57, and mAP at 0.5:0.95 of 0.21 for all three classes. The breakdown of Average Precision (AP) for our three classes in the validation set at an IoU threshold of 0.5, together with the related precision and recall values, are shown in Fig. 6.18. Below fig 6.19 shows the Confusion matrix and 6.20 shows F1 graph, precision-recall graph, recall graph, and precision graph respectively

Class	Images	Labels	p	R	mAP@.5	mAP@.5:.95:
all	343	398	0.76	0.51	0.574	0.211
Hole	343	80	0.707	0.362	0.408	0.139
Stitch	343	216	0.705	0.324	0.444	0.134
seam	343	102	0.868	0.843	0.868	0.358

250 epochs completed in 20.676 hours.

Figure 6.18: Results of YOLOv7-W6 Experiment

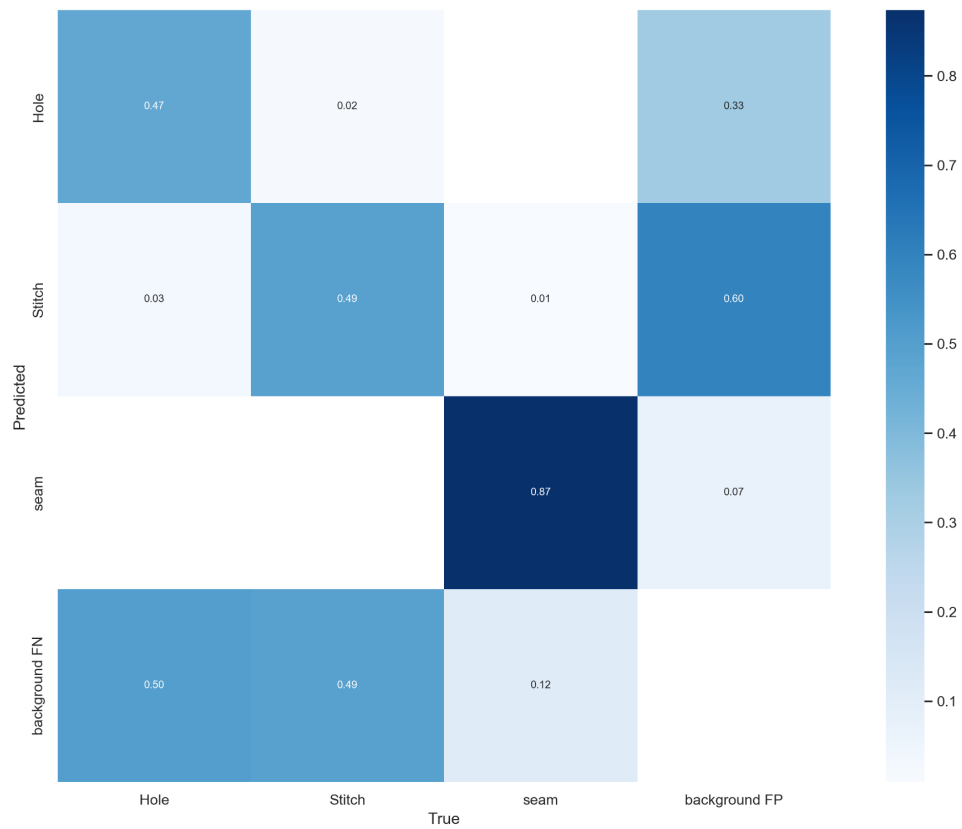


Figure 6.19: Confusion Matrix

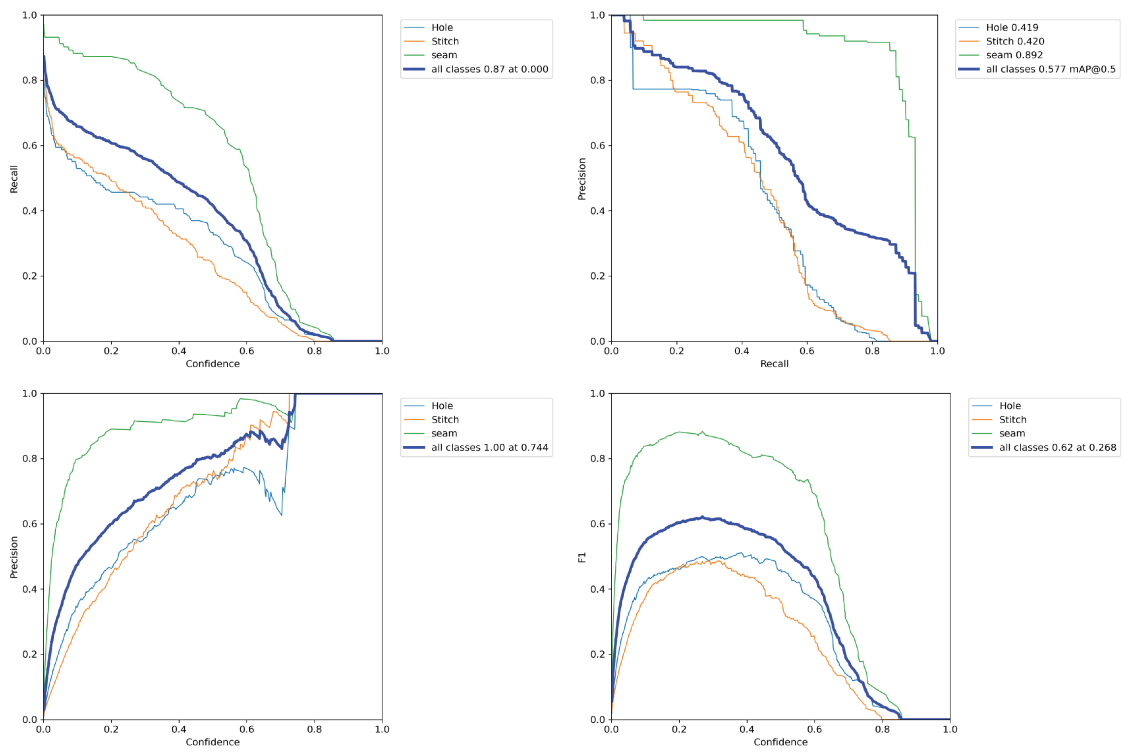


Figure 6.20: Recall graph, precision-recall graph, precision graph, and F1 graph

Chapter 7

Result Analysis

From the training evaluation, it seems that the YOLOv7 model with a learning rate of 0.001 is giving better accuracy than with a learning rate of 0.01. So in this part of the analysis, we are considering only the model with better accuracy of YOLOv7. For evaluation, both of the models run on NVIDIA GeForce RTX3080ti by using test.py and test dataset.

7.1 Result of YOLOv7

Class	Images	Labels	P	R	mAP@.5	mAP@.5:.95: 100%
all	362	473	0.71	0.594	0.611	0.255
Hole	362	138	0.523	0.406	0.379	0.163
Stitch	362	233	0.67	0.455	0.501	0.191
seam	362	102	0.936	0.922	0.952	0.412

Speed: 13.2/1.7/14.9 ms inference/NMS/total per 640x640 image at batch-size 32
Results saved to runs/test/yolov7 640 val5

Figure 7.1: Results of YOLOv7 Test set

After evaluating the test dataset Figure 7.1 shows that the average detection accuracy of all three classes by using the YOLOv7 model is 61% at mAP@0.5 and 25% at mAP@0.5:0.95. Where the seam class got the highest accuracy rate with 95%. And other two classes Hole and Stitch got 37% and 50% respectively. The inference speed here is 13.2 ms. Figure 7.2 shows detection on test images using the YOLOv7 model on three subclasses of Seam, Hole, and Stitches respectively.



Figure 7.2: Seam, Hole, and Stitch

Below fig 7.3 shows the Confusion matrix and 7.4 shows the F1 graph, precision-recall graph, Recall graph, precision graph respectively.

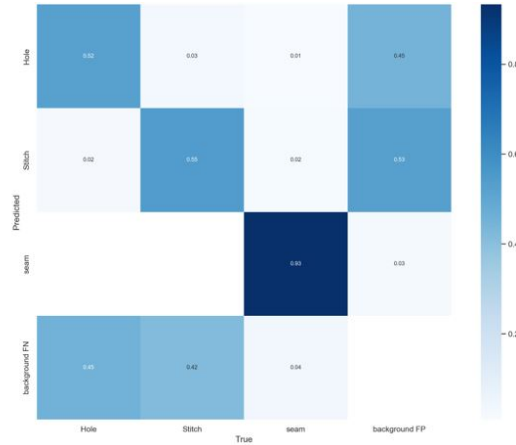


Figure 7.3: Confusion Matrix

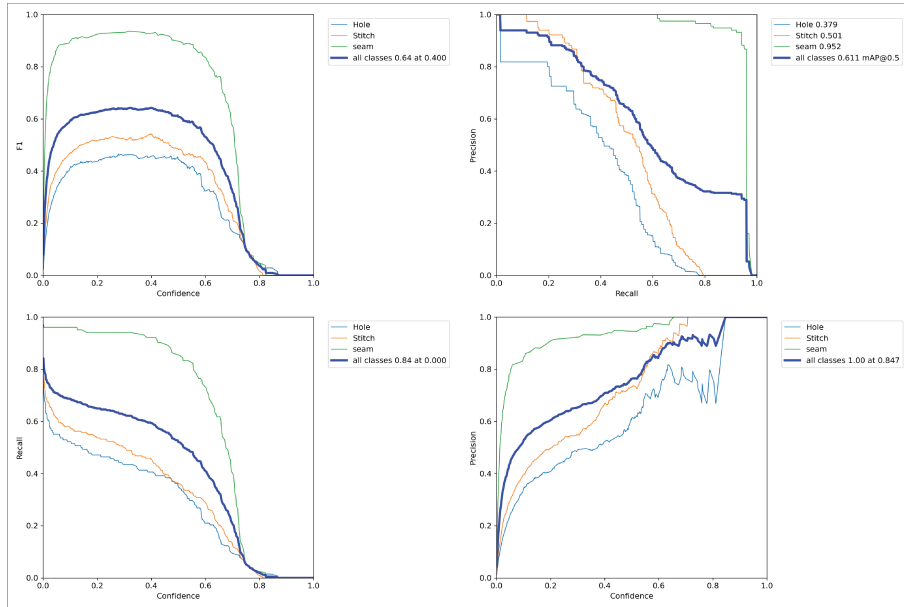


Figure 7.4: F1 graph, precision-recall graph, Recall graph, precision graph

7.2 Result of YOLOv7x

After evaluating on test dataset Figure 7.5 shows that the average detection accuracy of all three classes by using YOLOv7x model is 64% at mAP@0.5 and 25% at mAP@0.5:0.95. Where seam class got the highest number of accuracy rate with 97%. And other two classes Hole and Stitch got 44% and 50% respectively. The inference speed here is 21.5 ms. Figure 7.6 shows detection on test images using YOLOv7 model on three subclasses of seam, Hole and Stitches respectively.

Class	Images	Labels	P	R	mAP@.5	mAP@.5:.95: 100%
all	362	473	0.695	0.646	0.641	0.252
Hole	362	138	0.504	0.5	0.444	0.167
Stitch	362	233	0.63	0.496	0.509	0.195
seam	362	102	0.95	0.941	0.968	0.393

Speed: 21.5/1.4/22.9 ms inference/NMS/total per 640x640 image at batch-size 32
Results saved to runs/test/yolov7x

Figure 7.5: Results of YOLOv7x Experiment



Figure 7.6: Seam, Stitch and Hole

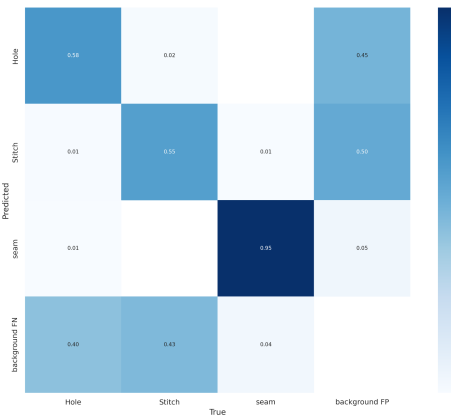


Figure 7.7: Confusion Matrix

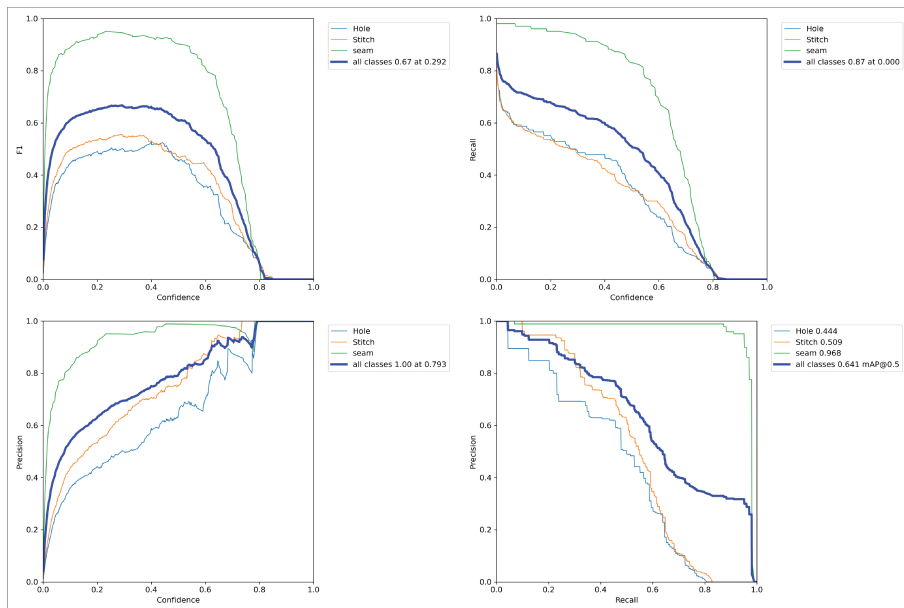


Figure 7.8: Recall graph, precision-recall graph, precision graph, and F1 graph

7.3 Result of YOLOv7-w6

After evaluating on test dataset Figure 7.9 shows that the average detection accuracy of all three classes by using YOLOv7-w6 model is 57% at mAP@0.5 and 20% at mAP@0.5:0.95. Where seam class got the highest number of accuracy rate with 89% .And other two classes Hole and Stitch got 40% and 52% respectively. The

Class	Images	Labels	P	R	mAP@.5	mAP@.5:.95
all	362	473	0.671	0.569	0.575	0.208
Hole	362	138	0.535	0.434	0.408	0.131
Stitch	362	233	0.571	0.429	0.424	0.151
seam	362	102	0.905	0.845	0.892	0.342

Speed: 5.2/0.9/6.1 ms inference/NMS/total per 640x640 image at batch-size 32
Results saved to runs\test\yolov7w6_best

Figure 7.9: Results of YOLOv7-w6 Experiment

inference speed here is 5.2 ms. Figure 7.10 shows detection on test images using YOLOv7-w6 model on three subclasses of Seam, Hole and Stitch respectively.



Figure 7.10: Hole, Seam and Stitch

Below fig 7.11 shows Confusion matrix and 7.12 shows recall graph, precision-recall graph, precision graph, and F1 graph respectively.

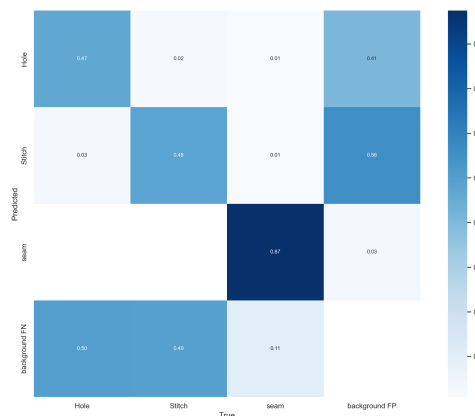


Figure 7.11: Confusion matrix

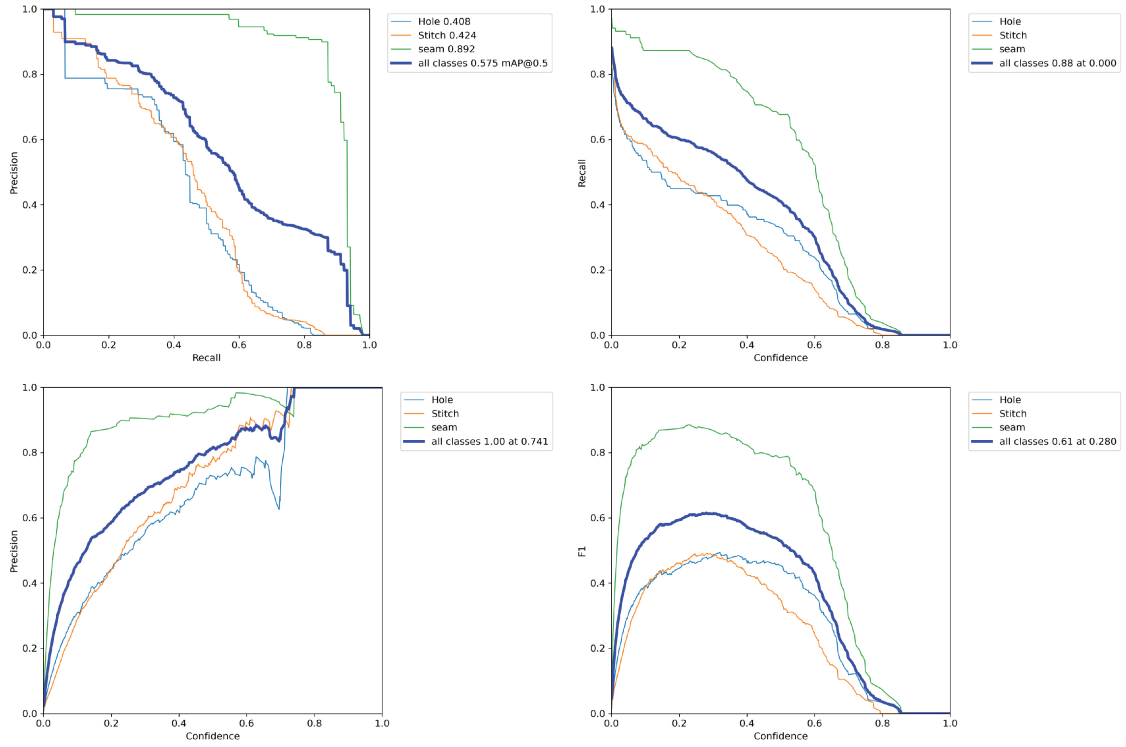


Figure 7.12: Recall graph, precision-recall graph, precision graph, and F1 graph

7.4 Real-time FPS calculation

In real-world applications, this kind of detection is done based on the video. For this, we used a recorded video and ran with both models best.pt file using detect.py on NVIDIA RTX3080ti. By analyzing the result YOLOv7 gives detection with a maximum FPS of 36 ms, YOLOv7x's maximum is 32 ms and lastly, YOLOv7-w6 gives a maximum FPS of 28.

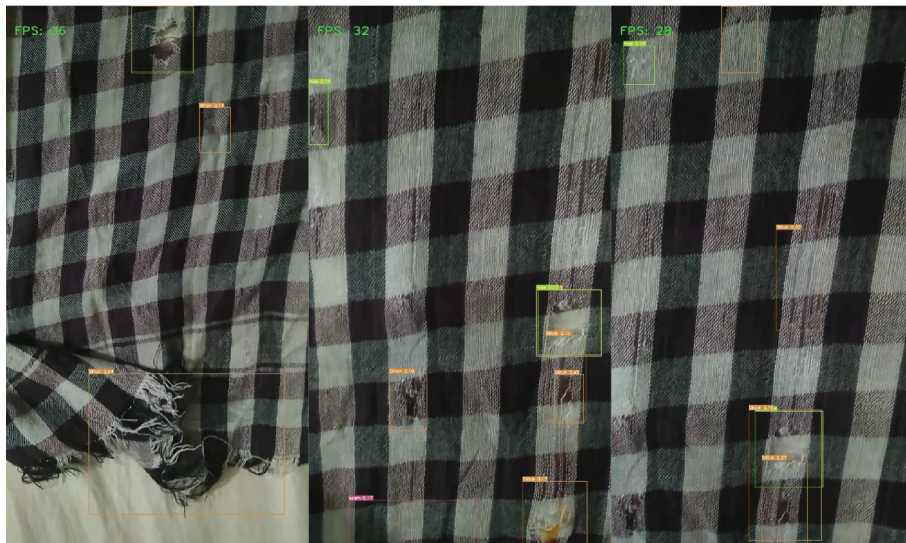


Figure 7.13: Real time Detection using YOLOv7, YOLOv7x, YOLOv7-w6

7.4.1 Comparison of Results

The results of the two models individually have previously been covered in great depth. We evaluate our three models one last time in this part based on how they performed on the test set of our dataset since this gives us an objective evaluation of the performance of our final models. The specifics are presented in Fig. 7.14 and we compare using mAP at 0.5 and mAP at 0.5:0.95. By analyzing all the parameters

Model Name	map@ 5	map@ .95	Precision	Recall	Maximum FPS(ms)
Yolov7	0.61	0.252	0.71	0.59	36
Yolov7x	0.64	0.25	0.69	0.65	32
YOLOv7-w6	0.57	0.2	0.67	0.56	28

Figure 7.14: Comparisons of YOLOv7, YOLOv7x and YOLOv7-w6

YOLOv7x comes up with better accuracy of 64% in contrast of YOLOv7's accuracy of 61% And YOLOv7-w6's AP of 57%. In terms of detection speed, YOLOv7 has the highest FPS of 36ms while YOLOv7x came up with 32ms while YOLOv7-w6 has the lowest FPS of 28ms.

Chapter 8

Conclusion

8.1 Conclusion

We present an overview of YOLOv7 real-time garment fault detection for the garment industry. The literature deems a manual human inspection process inadequate and ineffective for meeting the demands of the textile industry today. Computer vision and digital image processing may provide the foundation for meeting this industrial need. We typically use Pytorch and Convolutional Neural Networks, which can detect some but not all product flaws. YOLOv7 is one of a number of techniques that can be used to solve various problems. YOLOv7x is more accurate than YOLOv7 and YOLOv7-w6 in terms of increasing quality and lowering production costs of the finished textile product with a good AP and FPS. Our research will help in the future implementation in terms of choosing a model according to its requirements among YOLOv7, YOLOv7x, and YOLOv7-w6.

8.2 Future Scope

We used primary data-set in our research paper because it is difficult to find reliable data-set because they are not easily accessible online to all researchers because they are paid, and given the large number of fabric defect detection methods for which various databases, metrics, and imaging systems exist, the validity and reliability of methods are far from objectivity. To address this issue, we need to expand the clothing database in order to operate more efficiently. Also, implement our models in a more effective way in terms of production and this can change the scenario of the quality check process of our garments.

Bibliography

- [1] M. M. Islam, A. M. Khan, and M. M. Islam, “Textile industries in bangladesh and challenges of growth,” *Research Journal of Engineering Sciences*, vol. 2278, p. 9472, 2013.
- [2] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” *Cv-foundation.org*, pp. 779–788, 2016. [Online]. Available: https://www.cv-foundation.org/openaccess/content_cvpr_2016/html/Redmon_You_Only_Look_CVPR_2016_paper.html.
- [3] I. M. S, A. R. Rakib, and A. Adnan, *Ready-made garments sector of bangladesh: Its contribution and challenges towards development*, ResearchGate, Jun. 2016. [Online]. Available: <https://www.researchgate.net/publication/309702078-Ready-Made-Garments-Sector-of-Bangladesh-Its-Contribution-and-Challenges-towards-Development>.
- [4] M. Chablani, “Yolo, you only look once, real time object detection explained,” *Towards Data Science*, 2017.
- [5] J. Li, Z. Su, J. Geng, and Y. Yin, “Real-time detection of steel strip surface defects based on improved yolo detection network,” *IFAC-PapersOnLine*, vol. 51, pp. 76–81, 2018. DOI: 10.1016/j.ifacol.2018.09.412. (visited on 12/30/2019).
- [6] H.-w. Zhang, L.-j. Zhang, P.-f. Li, and D. Gu, “Yarn-dyed fabric defect detection with yolov2 based on deep convolution neural networks,” *2018 IEEE 7th Data Driven Control and Learning Systems Conference (DDCLS)*, May 2018. DOI: 10.1109/ddcls.2018.8516094. (visited on 07/16/2022).
- [7] A. Paszke, S. Gross, F. Massa, *et al.*, “Pytorch: An imperative style, high-performance deep learning library,” *Advances in neural information processing systems*, vol. 32, 2019.
- [8] M. Seçkin, A. Ç. Seçkin, and A. Coşkun, “Production fault simulation and forecasting from time series data with machine learning in glove textile industry,” *Journal of Engineered Fibers and Fabrics*, vol. 14, p. 1 558 925 019 883 462, 2019.
- [9] P. Bharati and A. Pramanik, “Deep learning techniques—r-cnn to mask r-cnn: A survey,” *Computational Intelligence in Pattern Recognition*, pp. 657–668, 2020.
- [10] J. Jing, D. Zhuo, H. Zhang, Y. Liang, and M. Zheng, “Fabric defect detection using the improved yolov3 model,” *Journal of Engineered Fibers and Fabrics*, vol. 15, p. 155 892 502 090 826, Jan. 2020. DOI: 10.1177/1558925020908268.

- [11] M. S. Biradar, B. Shiparamatti, and P. Patil, "Fabric defect detection using deep convolutional neural network," *Optical Memory and Neural Networks*, vol. 30, no. 3, pp. 250–256, 2021.
- [12] J. Choi, H. Y. Yeom, and Y. Kim, "Implementing cuda unified memory in the pytorch framework," in *2021 IEEE International Conference on Autonomic Computing and Self-Organizing Systems Companion (ACSOS-C)*, IEEE, 2021, pp. 20–25.
- [13] R. Jin and Q. Niu, "Automatic fabric defect detection based on an improved yolov5," *Mathematical Problems in Engineering*, vol. 2021, 2021.
- [14] Y. Wang, Z. Hao, F. Zuo, and S. Pan, "A fabric defect detection system based improved yolov5 detector," *Journal of Physics: Conference Series*, vol. 2010, p. 012191, Sep. 2021. DOI: 10.1088/1742-6596/2010/1/012191. (visited on 01/07/2022).
- [15] O. Chernytska, *Explainable defect detection using convolutional neural networks: Case study*, Medium, Aug. 2022. [Online]. Available: <https://towardsdatascience.com/explainable-defect-detection-using-convolutional-neural-networks-case-study-284e57337b59>.
- [16] H. Du, W. Zhu, K. Peng, and W. Li, "Improved high speed flame detection method based on yolov7," *Open Journal of Applied Sciences*, vol. 12, pp. 2004–2018, 2022. DOI: 10.4236/ojapps.2022.1212140. (visited on 01/06/2023).
- [17] C. Li, L. Li, H. Jiang, *et al.*, "Yolov6: A single-stage object detection framework for industrial applications," *arXiv:2209.02976 [cs]*, Sep. 2022. [Online]. Available: <https://arxiv.org/abs/2209.02976>.
- [18] Q. Lin, G. Ye, J. Wang, and H. Liu, "Roboflow: A data-centric workflow management system for developing ai-enhanced robots," in *Conference on Robot Learning*, PMLR, 2022, pp. 1789–1794.
- [19] A.-A. Tulbure, A.-A. Tulbure, and E.-H. Dulf, "A review on modern defect detection models using dcnn – deep convolutional neural networks," *Journal of Advanced Research*, vol. 35, pp. 33–48, Jan. 2022. DOI: 10.1016/j.jare.2021.03.015.
- [20] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," *arXiv:2207.02696 [cs]*, Jul. 2022. [Online]. Available: <https://arxiv.org/abs/2207.02696>.
- [21] N. D. T. Yung, W. K. Wong, F. H. Juwono, and Z. A. Sim, *Safety helmet detection using deep learning: Implementation and comparative study using yolov5, yolov6, and yolov7*, IEEE Xplore, Oct. 2022. DOI: 10.1109/GECOST55694.2022.10010490. [Online]. Available: <https://ieeexplore.ieee.org/abstract/document/10010490> (visited on 01/20/2023).
- [22] J. Zheng, H. Wu, H. Zhang, Z. Wang, and W. Xu, "Insulator-defect detection algorithm based on improved yolov7," *Sensors*, vol. 22, p. 8801, Nov. 2022. DOI: 10.3390/s22228801. (visited on 11/28/2022).