YOLOv8 and ResNet-50 Based Real-Time Fabric Defect Detection and Quality Grading System

Abdul Muchlis, Eri Prasetyo Wibowo, Rudi Irawan, Afzeri
Departement Information Technology, Gunadarma University, Depok, Indonesia,
Departement Mechanical Engineering, Gunadarma University, Depok, Indonesia
Department Mechanical Engineering, Indorama Engineering Polytechnic, Purwakarta, Indonesia
E-mail: muchlis07@staff.gunadarma.ac.id, eri@staff.gunadarma.ac.id, Rudi@staff.gunadarma.ac.id, afzeri.pei.ac.id

Keywords: YOLOv8, ResNet-50, fabric quality, automatic inspection, 4-Point System

Received: July 5, 2025

Automated fabric defect inspection is essential to address the low speed and inefficiency of manual inspection, which often requires machine stoppages to record defects. In contrast, this study presents a continuous, real-time defect detection and grading system based on the 4-point standard, integrating computer vision and deep learning techniques. A dataset of 1,456 annotated fabric images was developed from captured production videos. The proposed method employs ResNet-50 as the backbone for feature extraction and YOLOv8 for object detection, followed by defect quantification to determine fabric grades. Experimental evaluation shows that the model achieves a precision of 0.789, recall of 0.554, mAP50 of 0.675, mAP50-95 of 0.461, and an F1-score of 0.61. The class-wise mAP50 scores are 0.744 for line defects, 0.839 for stain defects, and 0.442 for hole defects. While the integration of convolutional neural networks with automated grading logic enhances detection accuracy and reliability in textile quality control, the relatively low performance on hole defects indicates the impact of dataset imbalance, as this category contained significantly fewer samples. Addressing this limitation through targeted data augmentation or additional data collection is recommended to improve generalization and achieve balanced detection performance across defect categories.

Povzetek: Članek obravnava problem počasnega in nezanesljivega ročnega pregleda tekstila. Predlaga metodo, ki združuje YOLOv8 z ogrodjem ResNet-50 za sprotno detekcijo napak ter samodejno razvrščanje kakovosti po 4-točkovnem sistemu. Model učinkovito zazna madeže in črte, a slabše luknje zaradi neuravnoteženega nabora podatkov.

1 Introduction

The integration of advanced information technologies particularly artificial intelligence (AI), machine learning (ML), and deep learning (DL) has fundamentally transformed various stages of the global textile manufacturing process. These technologies have enabled real-time defect detection, predictive maintenance of machinery, intelligent quality control, and end-to-end automation, thereby significantly enhancing operational efficiency, product consistency, and decision-making accuracy across the textile supply chain [1], [2], [3]. The application of this technology enables the machines to automatically detect defects in fabric, identify complex patterns, and improve product quality without human intervention [4]. Deep learning models, such as Convolutional Neural Networks (CNN), are widely used in the visual inspection to detect imperfections in fabrics with higher accuracy compared to the conventional methods, thereby helping to speed up the quality control processes and to increase the effectiveness of fabric defect detection. According to the Allied Market Research report, the market value of AI in the textile industry is expected to reach $4.8 billion by 2031, with a compound annual growth rate (CAGR) of 26.4% from 2022 to 2031, indicating a significant growth in the adoption of this technology.

The application of AI in the industry 4.0 and the smart manufacturing in the textile sector have transformed the traditional production processes by increasing efficiency, reducing errors and optimizing production workflows. AI-based technologies, such as machine learning, computer vision, and robotics, are being integrated into various stages of textile production, enabling the manufacturers to automate complex tasks, reduce waste, and increase productivity [5]

The textile industry has experienced a major transformation with the application of artificial intelligence (AI), especially in fabric defect detection. A fabric defect is a flaw in a material that degrades its quality, reducing its value and utility.

Once these defects are on the markets, they cannot be repaired or returned. Therefore, ensuring defect-free products is essential to produce high-quality goods [6]. Quality control plays an important role in ensuring customer satisfaction and reducing production costs [7], [8]. Traditional approaches to fabric inspection often rely on manual processes that are time-consuming and susceptible to human errors. This process involves visual observation by the operator which not only requires high concentration but is also difficult to maintain for long periods of time. As a result, errors in detecting fabric defects, such as holes, lines and stains, become common, which ultimately affects product quality.

The development of technological advances in the field of artificial intelligence, especially deep learning, has opened up new opportunities to improve the accuracy and efficiency of fabric defect detection. One appropriate technology is the YOLO (You Only Look Once) model. YOLO is designed to detect objects in one processing stage, so that it is able to provide fast detection results. With its detection speed, YOLO is very suitable for use in the textile industry which requires a fast and efficient inspection process [9], [10]. In addition, YOLO implementation can be integrated with other automation systems to create a solution that can reduce dependence on human labor and increase consistency in fabric defect inspection.

Detecting fabric defects with a YOLO-based approach method, the textile industry can optimize the quality control process and meet higher quality standards. This technology not only addresses the challenges of manual inspection but also represents a step forward towards digitalization and automation in the modern textile industry. Fabric defect detection using deep learning has been carried out using 2 methods in terms of the process, namely Double Shot Detector and Single Shot Detector [11]. The previous research using double shot detectors such as that conducted by research using a deep learning-based framework with a two-stage strategy, used the CNN Inception-V1 architecture to detect local defects and the LeNet-5 model to recognize the type of defect. This framework achieved 96% accuracy, 97% detection rate, and 6% false rate, demonstrating the great potential of deep learning in supporting the automation and precision of fabric defect detection [13]. The use of the YOLO and Single Shot Detector (SSD) architectures differs in that YOLO produces a set of object predictions across the entire images, while SSD produces object predictions at the various resolution levels [11]. In terms of speed, the YOLO architecture is faster than SSD but can provide better detection accuracy for objects of various scales and shapes [13]–[17].

Based on the previous research, it can be concluded that the development of deep learning technology has become the right step in detecting defects in fabrics. This technology is able to replace the role of human inspectors in the process of automatically detecting fabric defects, which ultimately increases the efficiency in terms of both time and production costs. By leveraging the advantages of deep learning [17], [18], [19], the textile industry can accelerate the fabric inspection process while ensuring higher accuracy, resulting in products with more consistent quality.

The 4-point system provides a way to determine defects according to their severity by assigning demerit or penalty holes. This system falls under ASTM D5430- 93, a standard test method for determining fabric grading [20]. This system is used by INDORAMA SYNTHETICS LTD. in conducting fabric defect inspections, although the weaving process is carried out in a modern way with automated weaving machines, there are still defects that occur, then an inspection is carried out by an inspector with a fabric length of 100 yards or 1 roll of fabric with a width of 1.8 meters.

Previous studies have generally focused on the application of deep learning models using image processing technology but have not yet reached the stage of fabric quality classification based on the 4-point system [19], In contrast, this study integrates both hardware and software components to develop an automated deep learning-based system capable of detecting and directly classifying fabric quality according to the 4-point grading standard. This integration can be seen in the research scheme in Figure 1, the fabric video was taken using an Intel RealSense D435i camera located above the fabric during the data acquisition process using a DC motor, then the data were processed into a dataset for the model training process. Once the model is good, the quality control process can be directly processed by integrating the system as in Figure 1.

Based on these issues, this study is focused on addressing two main research questions: (1) Can the YOLOv8 model combined with a ResNet-50 backbone detect fabric defects in real-time with sufficient accuracy? and (2) Can the proposed system automatically classify fabric quality using the 4-point system based on CNN predictions? These research questions provide a structured direction for system development and serve as the foundation for the evaluation framework in this study.

2 Literature review

Fabric defect detection is an important aspect of quality control in the textile industry. Automation in this process plays a vital role in improving the efficiency and consistency of an inspection. Convolutional Neural Network (CNN), especially the YOLO architecture, has been widely used for real-time object detection tasks due to its high accuracy and speed. Various previous studies have used the YOLO model to detect fabric defects, focusing on the model augmentation and modification techniques to improve the performances [21], [22].

2.1 Acquisition from video

The data collection process was carried out by recording the moving fabric using an Intel RealSense D435i camera. The video footage is then processed into individual image frames which are used to train the YOLOv8 model. In contrast to the static image datasets, this method allows a dynamic data acquisition, so that the fabric defects can be captured from the various fabric orientations and lighting conditions similar to the real-world conditions. The rotational and translational speeds of the fabric are precisely controlled to ensure a uniform coverage of the fabric area [23]. The fabric translational speed is 4.07 meters/minute.

2.2 Augmentation techniques

In 2022, Yue, Wang utilized the improved YOLO-based small target detection technology. This research uses data augmentation such as mosaic, impulse noise, Gaussian Blur, mirror flip, and affine transform, as well as anchor regrouping in the added dataset. Meanwhile, in this research, the imbalanced datasets were overcome by increasing and balancing the number of datasets in each class, and the augmentation techniques such as flipping, rotation, and contrast adjustment were applied. This augmentation is in line with the previous research [15], [21], where a similar method was proven to be effective in increasing the detection accuracy. The results show an increase in the small target detection accuracy of up to 8% with a mAP of 56.68% compared to the base model of 52.17%, especially in the hole and knot defects.

2.3 Deep learning model

Research on textile defect detection has advanced rapidly with the evolution of the *You Only Look Once* (YOLO) architecture. Wang improved YOLOv4 by incorporating SoftPool into the Spatial Pyramid Pooling (SPP) module and applying *contrast-limited adaptive histogram equalization* (CLAHE), achieving a 6% increase in mAP over the standard YOLOv4 [24]. In the same year, Wang combined YOLOv5 for defect detection with ResNet for classification in automotive *heat staking*, attaining a detection mAP of 95.1% and a classification F1-score of 98%, though at the cost of increased computational complexity due to its two-stage nature [25]

Later works shifted to optimizing YOLOv8. Li proposed FF-YOLO, a lightweight variant with Slim Attention Module (SimAM) and Lightweight Multi-Level Asymmetry Detector Head (LADH), which maintained high accuracy (mAP@0.5 of 75.5%) while reducing computational cost [26]. In 2025, DA-YOLOv8s integrated Deformable Convolution v2 (DCNv2) and Polarized Self-Attention (PSA), improving feature representation and achieving high-speed detection (257.38 FPS)[27].

This study extends these developments by embedding ResNet directly into YOLOv8 as its backbone, enabling *end-to-end* detection with deeper feature extraction from the earliest stages. This integration aims to improve accuracy for small or subtle textile defects while preserving real-time performance, bridging the gap between hybrid YOLO–ResNet pipelines and fully integrated, high-performance YOLO models.

Year	Study Title	Authors	Methodology	Key Results	Main Contribution
2022	A Fabric Defect Detection Method Based on Deep Learning	Wang et al.	YOLOv4 + SoftPool + CLAHE	mAP ↑ 6% vs. YOLOv4	Improved textile defect accuracy via enhanced feature pooling and preprocessing.
2022	Application of YOLO and ResNet in Heat Staking Process Inspection	Wang et al.	YOLOv5 + ResNet	mAP: 95.1%; F1: 98%	Combined fast YOLO detection with accurate ResNet classification for industrial inspection.
2024	FF-YOLO: Fashion Fabric Detection Algorithm Based on YOLOv8	Li et al.	YOLOv8 + SimAM + LADH	mAP@0.5: 75.5%; FLOPS ↓ 5%	Maintained high accuracy while reducing computational complexity.
2025	Textile Defect Detection Algorithm Based on the Improved YOLOv8	Song W, Lang D	YOLOv8 + DCNv2 + PSA	mAP@0.5: 44.6%; FPS: 257.38	Enhanced spatial channel feature representation for high-speed detection.
2025	**Proposed Model**		YOLOv8 + ResNet (backbone)	mAP@0.5: 67.5%	Integrated ResNet backbone for *end-to-end* deep feature extraction and real-time textile defect detection.

2.4 Four point system

The 4-Point system or four-point system is an approach often used in the textile industry to evaluate fabric quality based on detected defects. This system provides comprehensive metrics for assessing fabric condition by considering four key dimensions: defect density, defect distribution, defect size, and defect severity. This approach not only assists in a decision making in the production process but also becomes a widely accepted quality standard [28]. This assessment uses a penalty system in the form of a score from 1 to 4 based on the severity and size of the defects that occur. The application of the 4-point system by converting the accumulation of the number, size and severity of defects with total holes. These holes are used to classify the quality grade of the fabric, which is calculated based on the maximum length of the bounding box.

Formulation for calculating bounding box[26]–[28].

$$w, h = x_2 - x_1,\ y_2 - y_1 \tag{1}$$

Then the pixel length is obtained which is used to convert the actual size to pixels in the images using the DPI (Dots per Inch) formula which is shown in equation.

$$Pixel\ Length\ (cm) = \frac{Pixel\ length}{\mathrm{DPI}}\ x\ 2.54 \tag{2}$$

$$DPI = \frac{Pixel\ Widht}{Fabric\ Widht\ (inch)} \tag{3}$$

3 Method

Research on the development of a system of a fabric quality ranking device based on Deep Learning can be seen in Figure 1. The automation tool system is designed for taking images of moving fabric using an Intel RealSense D435i camera, producing a video which is then processed by a machine learning model to detect defects and determine fabric quality using a 4-point system. The research methodology is divided into 7 parts of the process, as follows:

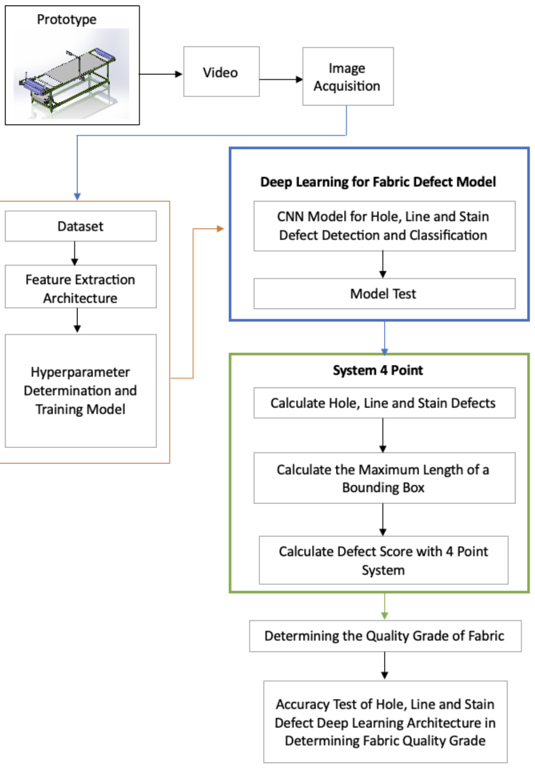

Prototype
Video
Image Acquisition
Dataset
Feature Extraction Architecture
Hyperparameter Determination and Training Model
Deep Learning for Fabric Defect Model
CNN Model for Hole, Line and Stain Defect Detection and Classification
Model Test
System 4 Point
Calculate Hole, Line and Stain Defects
Calculate the Maximum Length of a Bounding Box
Calculate Defect Score with 4 Point System
Determining the Quality Grade of Fabric
Accuracy Test of Hole, Line and Stain Defect Deep Learning Architecture in Determining Fabric Quality Grade

Figure 1: Framework's research

3.1 System design

Prototyping of fabric automation tools connected to an Intel RealSense D435i camera produces video and is processed for image acquisition.

3.2 Creating a classification model

3.2.1 Dataset acquisition

The image obtained from the camera acquisition with fabric from INDORAMA SYNTHETICS LTD. having been adjusted to the system produces video with a duration of 1.57 minutes converted into images producing 3,330 images. The 3,330 images were then manually selected by selecting images with hole, line and stain defects, resulting in 1,456 images which were used as datasets for the model training and testing processes. The images are augmented to make the dataset balanced in each defect class. The techniques used are as follows:

a) Flip: Horizontal and vertical
b) 90° Rotation:Clockwise and counterclockwise
c) Rotation 15° to - 15
d) Exposure Bounding Box: -10% to 10%
e) Bounding Box Brightness: -15% to 15%

3.2.2 Feature extraction architecture

Replacing the YOLOv8 backbone with ResNet-50 enhances feature representation, particularly for inputs with complex textures such as fabric defects. The ResNet-50 blocks are integrated before the neck module, allowing deeper semantic features to be propagated to the YOLOv8 detection head while maintaining inference efficiency, this can be seen in Figure 11. This architectural modification improves generalization, especially in detecting subtle texture anomalies, without compromising detection speed or overall performance.

3.2.3 Hyperparameters

Determination of Hyperparameters and Training the Determination of the hyperparameters used is:

a) Batch size 32
b) Learning Rate 0.001429
c) Epoch 50, 100, 150 and Early Stop
d) IoU 0.3

3.2 Detection and classification machine learning for hole, line and stain defects

Although at first glance Algorithm 1 may appear to reiterate standard YOLO detection procedures, it actually integrates essential post-processing stages that are critical to this study, particularly within the context of the textile industry. The algorithm not only counts the number of each defect type (holes, lines, and stains) but also calculates severity levels based on the 4-point fabric grading system, including the assignment of weight to specific defect categories. Moreover, it computes the maximum defect length by analyzing the bounding box coordinates, which serves as a key reference in the quantification and classification of fabric quality.

Algorithm 1: Defect detection and counting

```
Step 1: Input: Model (img, stream=True, iou = 0.3)
Step 2: Calculation hole, line, stain:
  classCount = class name:[] for class name in classNames
  Count hole = 0
  Count Line = 0
  Count stain = 0
Step 3: Count Defects
  if currentClass = 'Hole' then
    count_hole += 1
    total_score += 1
  else if currentClass = 'Line' then
    count_line += 1
    weight ← calc_weight(defect_length(cm))
    total_score += weight
  else if currentClass = 'Stain' then
    count_stain += 1
    weight ← calc_weight(defect_length(cm))
    total_score += weight
  end if
Step 4: Calculate Maximum Bounding Box Length
  w, h = x2 - x1, y2 - y1
  defect_length_pixel = max(w, h) = 0
```

4 Scheme research

The system of the Deep Learning-based fabric quality level detection automation tool begins with the initial step of building a system of the defect detection image capture hardware. The integration of the hardware systems can be seen in Figure 2,

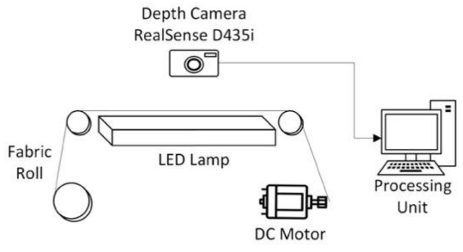


Figure 2: Scheme research

This scheme shows an automated fabric inspection system consisting of several main components. A RealSense D435i Depth Camera is used to capture precise images of the fabric. The camera is connected to a Processing Unit (computer) that processes the image data to detect the defects. The LED Lamp provides even lighting on the fabric to improve the accuracy of the camera's capture. Fabric Roll rolls the fabric, while DC Motor moves it continuously so that its entire surface can be inspected. This system is designed to automatically detect defects in fabric.

4.1 System design of automatic device capturing fabric images

The system device has dimensions and design as shown in Figure 3 which is used for video acquisition. The size or dimensions of the device have a device length of 100 cm and a device width of 30 cm and a device height of 25 cm. The camera position is perpendicular to the cloth with a height of 25 cm and the cloth is rolled at the same speed from the shaft driven by the electric motor, which is 4.07 meters / minute.

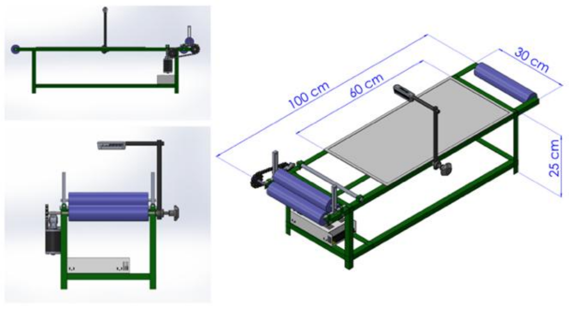


Figure 3: Device design

4.2 Manufacturing system of fabric image capturing device

The automatic image capturing device moves the fabric with a DC electric motor, while the fabric is rolled using a roller shaft. The camera is placed above the device while the fabric running to capture the fabric images and send them to the machine learning model for hole, line and stain defect detection and classification.

The sample of fabric obtained from INDORAMA SYNTHETICS LTD. is 5 meters long and 1.8 meters wide and it is adjusted to the system device which is 60 cm long and 30 cm wide. The results of the fabric adjusted to the system are 2 rolls of fabric of 15 meters each.

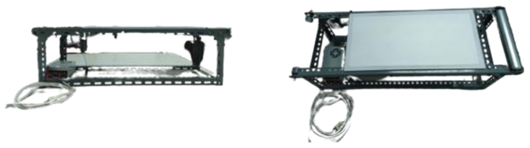


Figure 4: Aqquisition instrument

Table 1: Instrument design

Parameter	Value
Tool length	100 cm
Tool width	30 cm
Fabric board length	60 cm
Tool height	25 cm
Light width	30 cm
Sample Fabric width	1.8 meters = 180 cm
Sample Fabric length	5 meters
The ratio of fabric width to lamp width	180/30 = 6
Length of dataset fabric	0.6 m × 5 m = 30 m
Bearing type	Plain Bearing, diameter 40 mm
Power supply	220V to 12V 20A
Motor shaft diameter	9 mm
End cylinder shaft diameter	19 mm
Frame material	Angle Iron size 4×4 cm
RPM	35 revolutions in 1 minute
Time	3 minutes
Roll circumference	π × Roller diameter
Circumference	π × 3.7 cm = 11.64 cm

4.3 Video

During the fabric rolling process, video acquisition is performed using an Intel RealSense D435i camera mounted vertically above the fabric surface at a fixed height of 30 cm to ensure consistent top-down imaging. The resulting video has a duration of 1 minute 51 seconds with 30 frames per second.

4.4 Primary dataset

Prior to conducting the video analysis, a dataset was prepared using defective fabric samples provided by PT. INDORAMA. These fabric samples varied in length but consistently maintained a width of 1.8 meters. For systematic data acquisition, the fabric width was divided into six equal sections, each measuring 30 centimeters. These segmented widths were then reassembled longitudinally to form a continuous dataset. This configuration ensured consistent coverage and alignment of the defective areas along the entire fabric length.

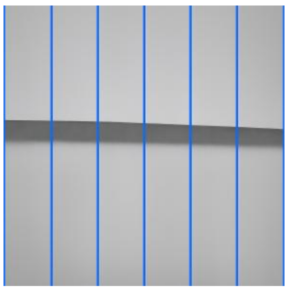
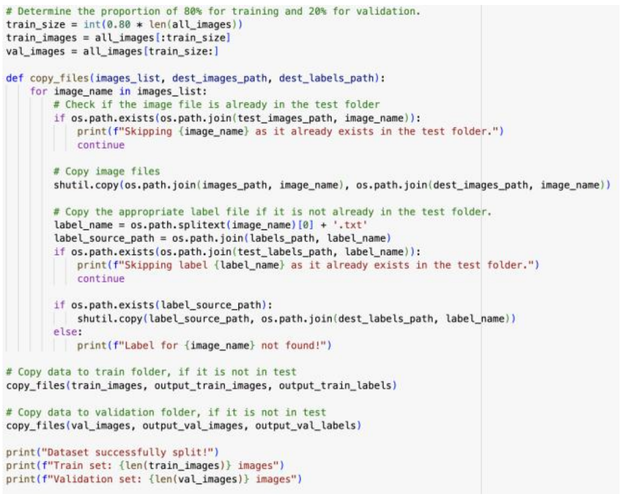


Figure 5: Symmetric sectioning of fabric image with indexed columns

Figure 5 illustrates the segmentation of fabric into six symmetrical vertical sections, each labeled sequentially from left to right. Following this segmentation, each section is vertically concatenated along the fabric's longitudinal axis to reconstruct a complete and continuous dataset image. This vertical merging process allows for a comprehensive and systematic analysis of defects distributed throughout the entire fabric surface the video results are frame extracted and converted from RGB format to Greyscale. The results of frame extraction from the video are the number of frames in the video of 3,330 images with a size of 1,568 pixels x 880 pixels.

4.5 Dataset preparation and processing

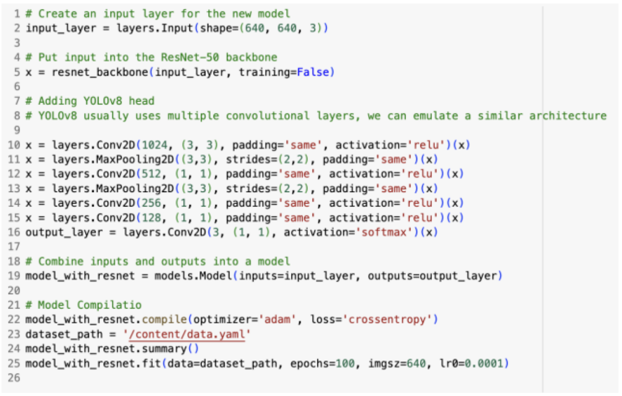
To improve the replicability of this study, we provide a detailed overview of the dataset preparation process. First, image data were captured by framing video segments from a real-time fabric inspection system. These images were then annotated using Roboflow, where preprocessing steps such as resizing, format conversion, and augmentation were applied as described in Subsection 3.2.1. The dataset was subsequently divided into training and validation sets using an 80:20 ratio through a custom Python script in Figure 6 ensuring no overlap with test data. For model testing, we used a different fabric roll than the one used in the training and validation process, ensuring independent evaluation.



```python
# Determine the proportion of 80% for training and 20% for validation.
train_size = int(0.80 * len(all_images))
train_images = all_images[:train_size]
val_images = all_images[train_size:]

def copy_files(images_list, dest_images_path, dest_labels_path):
    for image_name in images_list:
        # Check if the image file is already in the test folder
        if os.path.exists(os.path.join(test_images_path, image_name)):
            print(f"Skipping {image_name} as it already exists in the test folder.")
            continue

        # Copy image files
        shutil.copy(os.path.join(images_path, image_name), os.path.join(dest_images_path, image_name))

        # Copy the appropriate label file if it is not already in the test folder.
        label_name = os.path.splitext(image_name)[0] + '.txt'
        label_source_path = os.path.join(labels_path, label_name)
        if os.path.exists(os.path.join(test_labels_path, label_name)):
            print(f"Skipping label {label_name} as it already exists in the test folder.")
            continue

        if os.path.exists(label_source_path):
            shutil.copy(label_source_path, os.path.join(dest_labels_path, label_name))
        else:
            print(f"Label for {image_name} not found!")

# Copy data to train folder, if it is not in test
copy_files(train_images, output_train_images, output_train_labels)

# Copy data to validation folder, if it is not in test
copy_files(val_images, output_val_images, output_val_labels)

print("Dataset successfully split!")
print(f"Train set: {len(train_images)} images")
print(f"Validation set: {len(val_images)} images")
```

Figure 6: Split data

To enhance feature representation, particularly for subtle defect textures, the default YOLOv8 backbone was replaced with the pretrained ResNet-50 architecture. This modification leverages the residual learning capability of ResNet-50 to extract more discriminative and hierarchically rich features at the early stages of the network. Integration was performed by directly feeding the output of ResNet-50 into a series of convolutional layers designed to mimic the detection head of YOLOv8. These layers were configured to preserve spatial resolution and detection accuracy, especially for detecting small-scale defects such as holes and stains. This backbone replacement proved to yield more accurate bounding box predictions and clearer class separation, as reflected in the evaluation results on the validation data.



```python
# Create an input layer for the new model
input_layer = layers.Input(shape=(640, 640, 3))

# Put input into the ResNet-50 backbone
x = resnet_backbone(input_layer, training=False)

# Adding YOLOv8 head
# YOLOv8 usually uses multiple convolutional layers, we can emulate a similar architecture

x = layers.Conv2D(1024, (3, 3), padding='same', activation='relu')(x)
x = layers.MaxPooling2D((3,3), strides=(2,2), padding='same')(x)
x = layers.Conv2D(512, (1, 1), padding='same', activation='relu')(x)
x = layers.MaxPooling2D((3,3), strides=(2,2), padding='same')(x)
x = layers.Conv2D(256, (1, 1), padding='same', activation='relu')(x)
x = layers.Conv2D(128, (1, 1), padding='same', activation='relu')(x)
output_layer = layers.Conv2D(3, (1, 1), activation='softmax')(x)

# Combine inputs and outputs into a model
model_with_resnet = models.Model(inputs=input_layer, outputs=output_layer)

# Model Compilatio
model_with_resnet.compile(optimizer='adam', loss='crossentropy')
dataset_path = '/content/data.yaml'
model_with_resnet.summary()
model_with_resnet.fit(data=dataset_path, epochs=100, imgsz=640, lr0=0.0001)
```

Figure 7: YOLOv8 with ResNet-50 Backbone

This Figure 7 illustrates the architecture and training configuration of a deep learning model that combines the ResNet-50 backbone with a YOLOv8 detection head. The model is built using TensorFlow, starting with an input layer of 640 x 640 pixels, followed by multiple convolutional layers that emulate the YOLOv8 structure for object detection. The model is then compiled using the Adam optimizer and categorical crossentropy loss function, and trained for 100 epochs using a dataset in YAML format. This approach is designed to support efficient and automated fabric defect classification.

4.6 Classification model creation

Dataset acquisition is prepared before the data is processed in the learning model. The dataset images are selected manually from 3,330 existing images to become 1,456 bounding box images, divided into 80% training data, namely 1,165 images and 20% validation data, namely 291 images. The image size is resized to 640 pixels x 640 pixels in all dataset images.

The architecture can be seen in Figure 12 and explained as follows. The images pass through a convolution layer with Batch Normalization, ReLu activation function and Maxpooling, which reduces the image dimension to 320 x 320 with 64 features. The images go through several ResNet blocks to extract the features, with the first block producing an output of 160 x 160 with 256 features and the last block producing 20 x 20 with 2048 features, showing a decrease in dimension and an increase in feature depth. After the ResNet-50 backbone, down-sampling is performed through the convolution layer, reducing the features from 2,048 to

1,024, then the C2F block is applied to produce an output of 1024 features, followed by down-sampling to 512 features. The SPPF block then combines spatial information from multiple scales, producing an output of 1,024 features. The final detection layer predicts bounding boxes, objectness scores, and class labels, allowing the model to efficiently detect objects at multiple scales and produce accurate predictions.

ResNet is realized by adding skip connections between layers. These connections are element-by-element additions between the input and output blocks and do not add extra parameters or computational complexity to the network [23].

Determination and Training Epoch is used to see how many times the model sees the entire dataset during the training. The number of epochs is how many times the entire dataset is used to train the model. In each epoch, the model learns from the data repeatedly to improve the performance [30], [32]. The epochs used with 3 variations are epochs 50, 100 and 150. All three epochs were performed at a learning rate of 0.001429.

4.7 Machine learning detection and classification of hole, line and stain defects

a) CNN Model for Hole, Line and Stain Defect Detection and Classification
The model obtained from the classification model creation process is used to detect and classify the results of processing fabric image acquisition. The model results are detected images with an IOU value of 0.3 for the classification of 3 classes of fabric defects, namely lines, stain and holes. YOLO with its advantages can detect and classify objects [33].

b) Model test
A model testing uses the mean average precision (mAP) formula in the object detection by predicting the bounding box produced by the model compared to the ground truth box using an intersection over Union (IoU) [30], [31]. For each object class, the average precision (AP) value is calculated by taking the average of Precision over various Recall values. This AP is the main metric used to evaluate the performance of the model in detecting objects of a particular class. After AP is calculated for each class, mAP is calculated by taking the average of all these AP values. This mAP value reflects how well the model can detect objects, combining the information about precision and sensitivity in the object detection at various scales and classes, mAP provides a comprehensive picture of the model's performance in detecting the objects accurately and consistently across the datasets.

c) Calculation of Hole, Line and Stain Defects
The model can be used to detect and classify hole, line and stain defects. The next step is to calculate each defect. This calculation aims to process the calculation into a 4-point system which is an accumulation of the total defects that occur with the dimensions of the defects. Based on the total and dimensions of the defects, they are entered into a scale hole with a value of 1 to 4.

d) Calculation of the Maximum Length of the Bounding Box
Bounding box is an important element in the object detection in the form of hole, line and stain defect detection in the fabric images. Bounding box is calculated following the original size of the fabric to represent the size of the defect in the real image. The calculation that converts from pixels to the actual defect size uses DPI (Dots per Inch). The width w and height h values of the bounding box are calculated based on the difference in coordinates between the bottom-right and top-left holes of the bounding box. The max (w, h) function is then used to obtain the larger value between the width and height, which is called the maximum length of the bounding box in pixels. The function of converting pixel values to cm uses the formula (pixel length / DPI) x 2.54, where DPI (Dots Per Inch) is the scale factor used to determine the conversion from pixels to physical size [34].

e) Calculating Handicap Score With 4-Point System
The defect score according to the classification found is calculated using a 4-point system. Inspection in the research focuses on the inspection of the construction quality which is sourced from how the fabric itself is made. Fabric inspection is based on its production. With a 4-point system, defects in the warp/weft direction or strip/wale direction in a fabric will be given penalty holes using the criteria based on the American Society for Testing and Materials (ASTM) D5430-93 standard, as in Table 2.

Table 2: Penalty defect 4-point system

No	Defect Size	Penalty Point
1	Defect less than 3 inches	1
2	Defect more than 3 inches, but not more than 6 inches	2
3	Defect more than 6 inches, but not more than 9 inches	3
4	Defect more than 9 inches	4

f) Determining the Fabric Quality Level
Calculation of total holes per yard in the 4-point system, the quality of the fabric is evaluated in holes/100 square yards. The quality grade of the fabric is based on the number of holes against fabric defects with a length of Holes per 100·yard^2. This rule in attachment by INDORAMA SYNTHETICS LTD.

$$\text{Point} = \frac{\text{Total roll point x 6 x 100}}{\text{Fabric length (yard)x fabric weidht (inches)}} \quad (4)$$

The results of calculating the score using a 4-point system for fabric defects are used in the process of determining the fabric quality level (grade) by following the rules in Table 3.

Table 3: Fabric quality grade

No	Fabric Quality Grade	Total Defect
1	Grade AE	<30
2	Grade A	30 < A<40
3	Grade BE	>40

g) Accuracy Testing of Machine Learning Architecture for Detecting Hole, Line, and Stain Defects in Fabric Quality Grade

A system learning machine for detecting dot, line and soil defects in fabric has been completed and now requires testing to evaluate the accuracy of prediction and determination of fabric quality grade. The testing steps include calling the confusion_matrix (y_true, y_pred, labels) function to calculate the confusion matrix based on three main parameters, namely y_true (the actual label), y_pred (the label predicted by the model), and labels (the list of all classes). The calculation results of the function are stored in the variable cm, and the cm value is returned for further analysis.

5 Result

The Intel RealSense D435i camera captures fabric images in RGB format, generating video files. In this research, the recorded video has a duration of 1 minute and 51 seconds, with the automation tool operating at a speed of 4.07 meters per minute. The resulting video resolution is 1,568 × 880 pixels.

5.1 Results of classification model creation

The acquisition of an image dataset with a size of 640 pixels x 640 pixels is divided into 80% training data and 20% validation data from 1456 bounding box images. The distribution of the images in the training and validation processes can be seen in Figure 9.

Figure 8: Sample images

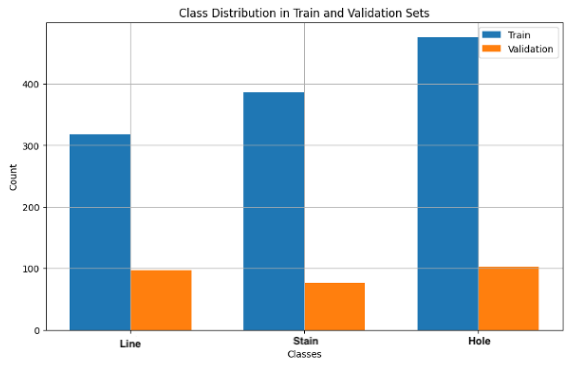


Figure 9: Distribution data

Table 4: Distribution results of training data and validation data

Class	Total Training Data	Total Validation Data	Total data	Training Data (%)	Validation Data (%)
Hole	476	102	578	82.4	17.6
Stain	386	77	463	83.4	16.6
Line	318	97	415	76.6	23.4

Table 4 shows the results of the distribution of training data and validation data on 3 defect classes, namely Hole, Line and Stain. Line defects, the number of training data used was 318 images, validation 97 images, the total image data used was 415 images, or 76.6% training data and 23.4% validation data if expressed in a percentage. Stain Defects the number of training data was 386 and validation 77, so that the total images used are 463, the percentage of training data is 83.4% and validation data is 16.6%. Hole Defects used 476 training data and 102 validation data so that the total is 578 so the percentage of training data is 82.4% and validation data is 17.6%.

Table 5: Impact of data split proportions on precision, recall, and mAP Values

Train Comp	Val. Comp	Precision	Recall	mAP	mAP50-95
70%	30%	0.718	0.647	0.705	0.462
80%	20%	0.729	0.646	0.704	0.464
90%	10%	0.743	0.562	0.658	0.413

The results presented in Table 5 demonstrate that variations in the proportion of training and validation data significantly influence the model's performance. As the proportion of training data increases from 70% to 90%, precision improves steadily (from 0.718 to 0.743), indicating the model's growing confidence in its predictions.

However, this comes at the cost of recall, which declines from 0.647 to 0.562, suggesting that the model becomes increasingly conservative and may fail to detect some true positives. Similarly, while mAP50 remains relatively stable between the 70:30 and 80:20 splits (0.705 and 0.704), it drops to 0.658 in the 90:10 split, implying reduced generalization when the validation set becomes too small. Notably, mAP50-95 shows a consistent downward trend as the validation portion decreases, from 0.462 at 70:30 to 0.413 at 90:10. These findings suggest that the 80:20 split achieves the best trade-off between precision, recall, and generalization. Conversely, the 70:30 split may be more appropriate when prioritizing recall and broader generalization performance.

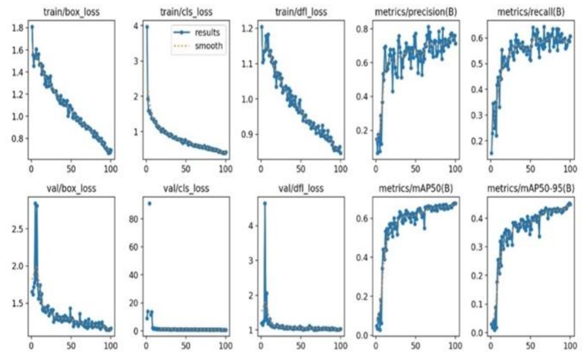


Figure 10: Performance graph at epoch 100

Figure 10 provides a comprehensive visualization of the model’s training and validation behavior over 100 epochs. The loss curves covering box loss, classification loss, and DFL (distribution focal loss) exhibit a consistent and smooth downward trend in both training and validation phases, indicating effective convergence and stability of the learning process. The sharp decline in validation losses during the initial epochs reflects rapid learning, while the subsequent plateau suggests the model has reached an optimal learning state with minimal overfitting. The performance metrics precision, recall, mAP50, and mAP50-95 all show a gradual improvement, particularly precision and mAP50, which stabilize above 0.75 and 0.70, respectively, by the end of training. However, recall and mAP50-95 demonstrate more fluctuation, suggesting sensitivity to class imbalance and the presence of harder-to-detect defect types. This graphical evidence reinforces earlier tabular analysis, validating that the model's performance peaks around epoch 100 with the 80:20 data split, achieving strong precision while balancing generalization. The curve patterns also highlight that extending training beyond 100 epochs may not yield significant gains, thus confirming epoch 100 as an optimal stopping point.

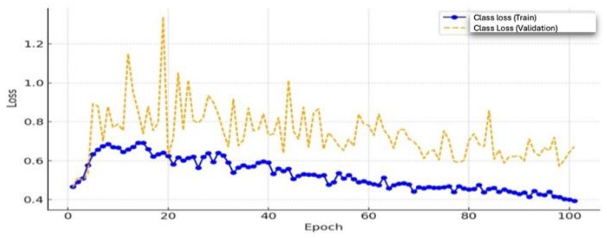
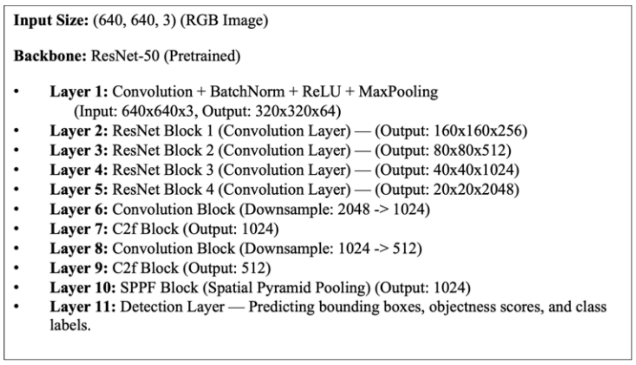


Figure 11: Loss train and valid graphs

Figure 11 shows the training and validation class loss curves. The training loss consistently decreases across epochs, indicating effective learning and convergence. However, the validation loss fluctuates with noticeable peaks, suggesting difficulty in generalization possibly due to data distribution differences or early signs of overfitting. The consistently higher validation loss compared to training loss also reflects a performance gap. This instability indicates the model has not yet fully generalized and could still be improved.

5.2 Architectural results of feature extraction

This architecture processes 640x640 pixel RGB fabric images using a ResNet-50 backbone [35]. The input passes through an initial convolutional layer and ResNet blocks 1-4 to extract features at various scales, with output dimensions ranging from 320x320x64 to 20x20x2048. The features are compacted through the convolution and Cross Stage Partial (C2F) blocks, followed by Spatial Pyramid Pooling (SPPF) to fuse the features from different scales. The final detection layer predicts bounding boxes, objectness scores and class labels, producing ready-to-use features for the training fabric defect detection models such as holes, lines, and stains.


Input Size: (640, 640, 3) (RGB Image)

Backbone: ResNet-50 (Pretrained)

- **Layer 1:** Convolution + BatchNorm + ReLU + MaxPooling
  (Input: 640x640x3, Output: 320x320x64)
- **Layer 2:** ResNet Block 1 (Convolution Layer) — (Output: 160x160x256)
- **Layer 3:** ResNet Block 2 (Convolution Layer) — (Output: 80x80x512)
- **Layer 4:** ResNet Block 3 (Convolution Layer) — (Output: 40x40x1024)
- **Layer 5:** ResNet Block 4 (Convolution Layer) — (Output: 20x20x2048)
- **Layer 6:** Convolution Block (Downsample: 2048 -> 1024)
- **Layer 7:** C2f Block (Output: 1024)
- **Layer 8:** Convolution Block (Downsample: 1024 -> 512)
- **Layer 9:** C2f Block (Output: 512)
- **Layer 10:** SPPF Block (Spatial Pyramid Pooling) (Output: 1024)
- **Layer 11:** Detection Layer — Predicting bounding boxes, objectness scores, and class labels.

Figure 12: Feature extraction architecture

5.3 Results of hyperparameter determination

The hyperparameters applied use the epoch trial values of 50, 100, 150 and early stopping so that it can be used to choose a good combination between epoch and learning rate 0.001429 and iou 0.3. The hyperparameter results in Table 6 show the model performance which can be seen in the precision and recall values.

Table 6: Hyperparameter experiment result

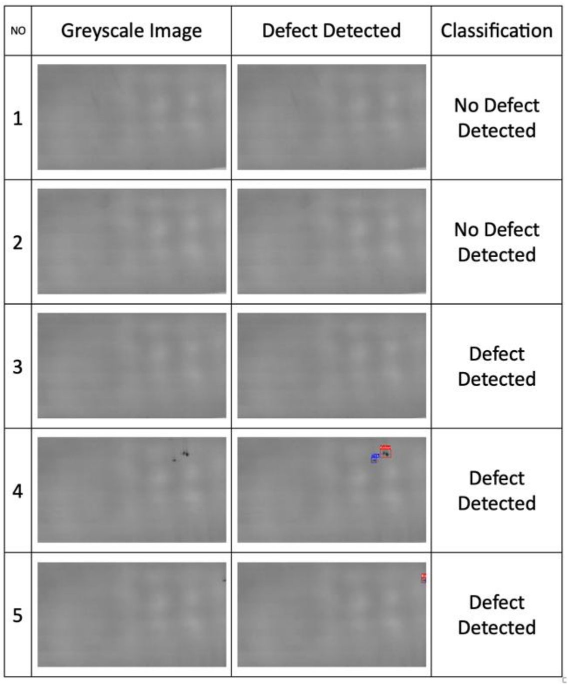
Epoch	Learning Rate	Precision	Recall	mAP 50	mAP 50-95
50	0.001429	0.701	0.632	0.671	0.44
100	0.001429	0.813	0.577	0.686	0.439
150	0.001429	0.702	0.63	0.664	0.413
Early stop at (67)	0.001429	0.720	0.574	0.663	0.413

The hyperparameter experiment results presented in Table 6 indicate a non-monotonic relationship between the number of training epochs and key evaluation metrics, such as precision, recall, and mAP. While the precision significantly increased from 0.701 at 50 epochs to 0.813 at 100 epochs, this gain was accompanied by a notable decline in recall (0.577) and minimal improvements in mAP50 (0.686) and mAP50-95 (0.439). A further increase to 150 epochs led to a regression in precision and map scores, suggesting the onset of overfitting, particularly when training is prolonged without adequate regularization mechanisms.

More critically, the downward trend in recall and the inconsistent mAP50-95 values can be attributed to the underrepresentation of the 'hole' defect class in the training data. This class imbalance hinders the model's ability to generalize well across all defect types, especially for small-scale or visually subtle defects such as holes, which are often harder to localize due to their limited spatial footprint and low texture contrast. The consistently low recall scores, even with increased epochs, reflect the model's bias toward the majority classes (e.g., lines and stains), which dominate the training set. This asymmetry in class representation has a compounding effect: while the model becomes more precise (fewer false positives), it fails to detect minority class instances, leading to false negatives that are critical in high-stakes quality assessment applications. In the context of fabric inspection, misclassification or omission of small yet significant defects like holes could result in unacceptable quality decisions downstream.

5.4 Machine learning results of detection and classification of hole, line and stain defects

The results of the process of calling the model and detecting defects in the images are the steps in the CNN model for detecting and classifying hole, line and stain defects can be seen in Figure 13.



NO	Greyscale Image	Defect Detected	Classification
1			No Defect Detected
2			No Defect Detected
3			Defect Detected
4			Defect Detected
5			Defect Detected

Figure 13: Detection model results

Model testing is done to test the model in recognizing defective objects and recognizing the types of fabric defects. The defects are in the form of holes, lines and stain.

Table 7: Model testing result on proposed model

Class	Precision	Recall	mAP50	mAP 50-95	F1-Score
All	0.789	0.554	0.675	0.461	0.61
Line	0.781	0.639	0.744	0.536	0.66
Stain	0.829	0.720	0.839	0.647	0.62
Hole	0.758	0.302	0.442	0.201	0.52

Table 7 presents the model testing results for three defect types: line, stain, and hole. Overall, the model achieved a precision of 0.789 and recall of 0.554, with a mAP50 of 0.675 and mAP50-95 of 0.461. The highest performance was observed in the *stain* class, with a precision of 0.829 and mAP50 reaching 0.839, indicating strong detection capability for stain defects. In contrast, the lowest performance was seen in the *hole* class, where recall dropped to 0.302 and mAP50-95 was only 0.201, suggesting that the model struggles to detect hole-type defects effectively.

Table 8: Model testing result on YOLOv8 Basic

Class	Precision	Recall	mAP50	mAP 50-95	F1-Score
All	0.652	0.421	0.531	0.336	0.51
Line	0.643	0.498	0.591	0.412	0.56
Stain	0.699	0.578	0.682	0.489	0.63
Hole	0.621	0.218	0.357	0.161	0.32

Based Table 7 and 8 compared to the YOLOv8 Basic model with the Proposed Model shows notable improvements across all evaluation metrics. Overall precision increased from 0.652 to 0.789, and recall improved from 0.421 to 0.554. The mAP50 and mAP50-95 also improved significantly, from 0.531 and 0.336 in YOLOv8 Basic to 0.675 and 0.461 in the Proposed Model, respectively.

The hole class, while still the most challenging, saw modest gains in mAP50 (from 0.357 to 0.442) and mAP50-95 (from 0.161 to 0.201). However, its recall remained low in both models, indicating that further enhancement is needed to detect hole-type defects effectively.

Although the Proposed Model demonstrates overall performance improvement compared to the YOLOv8 Basic model, the results for the hole class remain relatively low, particularly in terms of recall and mAP50-95. This indicates that the model struggles to consistently detect most hole-type defects. The low recall is likely due to the limited number of training samples and the high variability in the shape, size, and position of the defects, which constrain the model's ability to generalize effectively. This issue persists despite the application of data augmentation techniques intended to balance the distribution across defect types. Therefore, further strategies such as improving annotation quality, increasing real-world data for minority classes, or applying techniques like focal loss may be necessary to enhance the model's sensitivity to underrepresented defect categories.

Based on mAP@0.5 performance, the proposed model achieves a score of 0.675, representing a significant improvement over DA-YOLOv8 (0.446) and confirming the effectiveness of incorporating ResNet as the backbone in enhancing feature representation. Although it does not outperform FF-YOLO (0.755) or YOLOv5 + ResNet (0.951), the proposed model offers a strong balance between detection accuracy and computational efficiency. The YOLOv5 + ResNet architecture, despite its high accuracy, relies on a two-stage detection process that may introduce latency and increased complexity, making it less suitable for real-time applications. In contrast, the proposed model retains a single-stage detection framework while leveraging the enriched semantic features provided by the ResNet backbone, enabling more accurate object localization and classification without sacrificing inference speed. These results indicate that the proposed model is a promising solution for industrial settings that require both high precision and real-time performance.

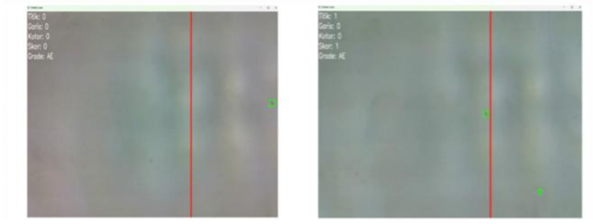
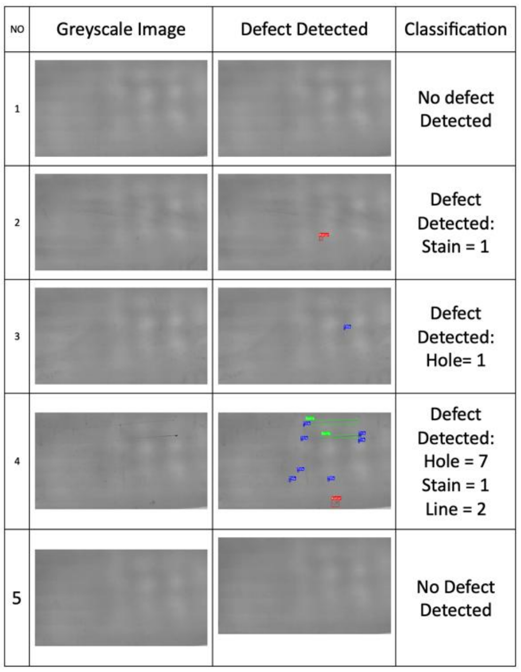


Figure 14: Objects before (left) and after (right) crossing the threshold

The hole, line and stain defect calculations are carried out after the detected object passes the bias line, so that they are calculated according to the defect classification of the object. Figure 14 (a) the object is detected before passing the bias line and all the assessment elements still have a value of 0. Figure 14 (b) shows that the detected object has passed the bias line, so that the assessment elements have changed according to the classification of the detected object.



NO	Greyscale Image	Defect Detected	Classification
1			No defect Detected
2			Defect Detected: Stain = 1
3			Defect Detected: Hole= 1
4			Defect Detected: Hole = 7 Stain = 1 Line = 2
5			No Defect Detected

Figure 15: Computation of defect detection based on classification

5.5 Calculation results of maximum bounding box length

The bounding box result is a box used to mark the detected objects. Figure 16 shows the object detection with a bounding box, the object can be detected by issuing the maximum bounding box length value.

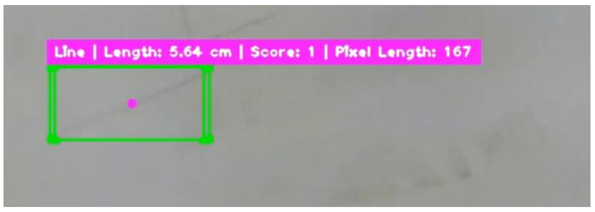


Figure 16: Bounding box display showing maximum length

The calculation of pixels to cm is as follows: There is a Bounding box with a pixel length of 167 Count Pixels to cm.

The bounding box result is a box used to mark the detected objects. Figure 16 shows the already object detection with a bounding box. The object can be detected by issuing the maximum length value of the bounding box. The calculation of pixels to cm is as follows:

1. There is a bounding box with a pixel length of 167.
2. Calculate Pixels to cm
3. Defect Length Result
4. The score obtained is 1 because it only meets the score limit of 1 to 8 cm.

5.7 Results of calculating defect score with 4-point system

The calculation of the defect scores is based on the number of defects in each class with the dimensions of the defects according to the categories listed in Table 2. Every time a defect passes the threshold, it is locked and the defect score is calculated based on the bounding box dimensions. The process can be seen in Figure 17.

5.8 Results of fabric quality determination

The fabric quality ranking (grade) based on a 4-point system has been successfully carried out by a system device integrated with a learning model for detecting hole, line and stain defects.

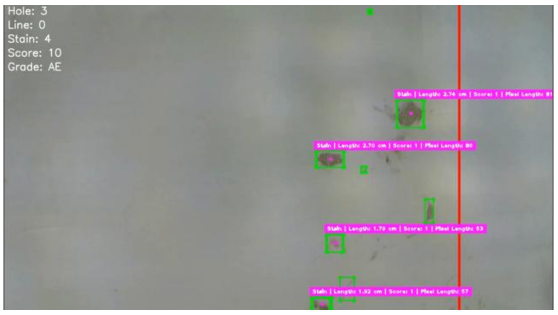

Hole: 3
Line: 0
Stain: 4
Score: 10
Grade: AE
Stain | Length: 2.74 cm | Score: 1 | Pixel Length: 81
Stain | Length: 2.70 cm | Score: 1 | Pixel Length: 80
Stain | Length: 1.79 cm | Score: 1 | Pixel Length: 53
Stain | Length: 1.92 cm | Score: 1 | Pixel Length: 57

Figure 17: Defect score calculation view

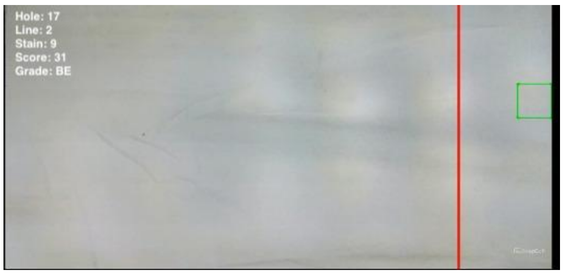


Figure 18: Results of defect detection interface and fabric quality grading

The fabric that runs on the automation tool and is captured by the camera into a frame is detected, classified and calculated for each defect found according to the type of defect. The rules used to determine the quality of the fabric use a 4-point system. The Grade result based on Figure 18 is BE.

5.9 Accuracy test results of learning machine architecture for hole, line and stain defects

The performance results of the proposed model, as summarized in Table 7 and further illustrated in the confusion matrix in Figure 19, provide critical insight into its classification capabilities across the three fabric defect categories: Line, Stain, and Hole. The aggregated performance metrics indicate that the model achieved a precision of 0.789, recall of 0.554, mAP50 of 0.675, mAP50-95 of 0.461, and an F1-score of 0.61. These values offer a coherent and consistent summary, resolving previously reported discrepancies in overall performance statistics. Class-wise, the model exhibited high confidence in detecting Line defects, as evidenced by 24 true positives and a competitive F1-score of 0.66. For the Stain class, the model achieved the highest recall (0.720) and mAP50 (0.839), albeit with notable misclassification into the Line class. The Hole class, however, posed the greatest challenge, with the lowest recall (0.302) and F1-score (0.52), suggesting difficulties in distinguishing its features due to potential class imbalance or inadequate discriminatory patterns. These findings highlight the need for data balancing and enhanced feature engineering, particularly for underrepresented classes, to improve the model's robustness and ensure consistent performance across all defect categories.

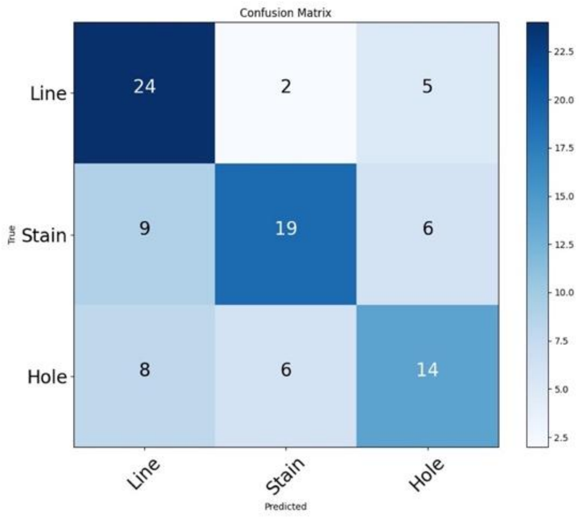


Figure 19: Results of confusion matrix

The analysis based on Figure 19 has good a model performance for the "Line" class with a few classification errors, while the performance in the "Stain" and "Holes" classes needs to be improved, especially to reduce errors between classes that have similar features.

6 Conclusion

This study successfully developed an automated system integrating the Intel RealSense D435i camera and a Convolutional Neural Network (CNN) model to detect and classify three types of fabric defects hole, line, and stain while simultaneously evaluating fabric quality based on the 4-point grading system. The hardware system, built with dimensions of 100 cm × 30 cm × 25 cm, enables real-time defect detection. The implemented CNN model demonstrated competitive performance, achieving a precision of 0.789, recall of 0.554, mAP50 of 0.675, mAP50-95 of 0.461, and an F1-score of 0.61. These results indicate a reasonable balance between classification accuracy and generalization capability on unseen data.

The primary advantage of this research lies in the seamless integration of hardware and software components, enabling the system to operate fully autonomously without human intervention. Notably, the implementation of the 4-point grading system represents a significant contribution, as it allows for the automated calculation of defect scores and determination of fabric grade without the need to stop the fabric rolling machine. This marks a substantial improvement over previous approaches that required manual inspection and temporary halting of the production process to perform grading. As a result, the proposed approach not only enhances inspection speed and classification accuracy but also significantly reduces production downtime, improves operational efficiency, and supports the automation of quality control in textile manufacturing.

Despite its promising results, the system presents several limitations. The training dataset is relatively limited in size and diversity, which may affect the model's ability to generalize to more complex or varied defect patterns. Additionally, the system has not yet been tested under real industrial conditions, leaving uncertainties regarding its stability and reliability in actual production environments.

Future research should focus on expanding the dataset with more diverse and representative defect types, employing data augmentation techniques to enrich model robustness, and conducting field trials in industrial settings to validate the system's performance under real-world operational constraints.

In summary, the automated system developed in this study not only demonstrates strong potential in improving the accuracy and speed of fabric quality inspection but also provides a scalable and practical solution for advancing quality control processes within the modern textile industry.

7 Acknowledgment

The authors would like to express their deepest gratitude to the Rector and Vice Rector II of Gunadarma University for their support has been given, especially in financing the publication of this research. The support and contribution given are very meaningful in realizing this work. Their sincere gratitude also goes to INDORAMA SYNTHETICS LTD. who has provided samples of fabric defects.

References

[1] M. Ateeq, A. P. P. Abdul Majeed, H. Hafizh, M. A. Mohd Razman, I. Mohd Khairuddin, and N. H. Noordin, "A Feature-Based Transfer Learning Method for Surface Defect Detection in Smart Manufacturing," in *Intelligent Manufacturing and Mechatronics*, W. H. Mohd. Isa, I. Mohd. Khairuddin, Mohd. A. Mohd. Razman, S. 'Atifah Saruchi, S.-H. Teh, and P. Liu, Eds., Singapore: Springer Nature Singapore, 2024, pp. 455–461.

[2] K. Gopalakrishnan and P. T. Vanathi, "Fabric Defect Detection Using Deep Learning Techniques," in *Ubiquitous Intelligent Systems*, P. Karuppusamy, F. P. García Márquez, and T. N. Nguyen, Eds., Singapore: Springer Nature Singapore, 2022, pp. 101–113.

[3] Q. Dong, "Surface defect detection algorithm for aluminum profiles based on deep learning," *Informatica*, vol. 48, no. 13, pp. 1–14, 2024.

[4] F. K. Konstantinidis, I. Kansizoglou, K. A. Tsintotas, S. G. Mouroutsos, and A. Gasteratos, "The role of machine vision in industry 4.0: A textile manufacturing perspective," in *IST 2021 - IEEE International Conference on Imaging Systems and Techniques, Proceedings*, 2021. doi: 10.1109/IST50367.2021.9651459.

[5] M. P. Sikka, A. Sarkar, and S. Garg, "Artificial intelligence (AI) in textile industry operational modernization," 2024. doi: 10.1108/RJTA-04-2021-0046.

[6] Y. A. Karayiannis *et al.*, "Defect detection and classification on web textile fabric using multiresolution decomposition and neural networks," in *Proceedings of the IEEE International Conference on Electronics, Circuits, and Systems*, 1999. doi: 10.1109/ICECS.1999.813221.

[7] E. Shady, Y. Gowayed, M. Abouiiana, S. Youssef, and C. Pastore, "Detection and Classification of Defects in Knitted Fabric Structures," *Textile Research Journal*, vol. 76, no. 4, 2006, doi: 10.1177/0040517506053906.

[8] C. H. Chan, "Fabric defect detection by Fourier analysis," *IEEE Trans Ind Appl*, vol. 36, no. 5, 2000, doi: 10.1109/28.871274.

[9] S. Das, M. K. Selvaraj, S. Shanmugavelu, and S. Jayaram, "Deep learning neural networks for knitted fabric defect identification and classification," *Man-Made Textiles in India*, vol. 50, no. 12, 2022.

[10] L. Shi, J. Song, Y. Gao, G. Cheng, and B. Hao, "YOLO-GFD: A Fast and Accurate Fabric Defect Detection Model," in *2023 4th International Conference on Big Data, Artificial Intelligence and Internet of Things Engineering, ICBAIE 2023*, 2023. doi: 10.1109/ICBAIE59714.2023.10281292.

[11] M. Li, D. Pi, and S. Qin, "An efficient single shot detector with weight-based feature fusion for small object detection," *Sci Rep*, vol. 13, no. 1, p. 9883, 2023, doi: 10.1038/s41598-023-36972-x.

[12] S. N. Appe, G. Arulselvi, and G. N. Balaji, "CAM-YOLO: tomato detection and classification based on improved YOLOv5 using combining attention mechanism," *PeerJ Comput Sci*, vol. 9, 2023, doi: 10.7717/peerj-cs.1463.

[13] Y. Liu, P. Sun, N. Wergeles, and Y. Shang, "A survey and performance evaluation of deep learning methods for small object detection," *Expert Syst Appl*, vol. 172, 2021, doi: 10.1016/j.eswa.2021.114602.

[14] N. D. Nguyen, T. Do, T. D. Ngo, and D. D. Le, "An Evaluation of Deep Learning Methods for Small Object Detection," *Journal of Electrical and Computer Engineering*, 2020, doi: 10.1155/2020/3189691.

[15] B. Liu *et al.*, "PRC-Light YOLO: An Efficient Lightweight Model for Fabric Defect Detection," *Applied Sciences (Switzerland)*, vol. 14, no. 2, 2024, doi: 10.3390/app14020938.

[16] X. Jun, J. Wang, J. Zhou, S. Meng, R. Pan, and W. Gao, "Fabric defect detection based on a deep convolutional neural network using a two-stage strategy," *Textile Research Journal*, vol. 91, no. 1–2, 2021, doi: 10.1177/0040517520935984.

[17] P. Bory, "Deep new: The shifting narratives of artificial intelligence from Deep Blue to AlphaGo," *Convergence*, vol. 25, no. 4, 2019, doi: 10.1177/1354856519829679.

[18] A. Zahra, M. Amin, F. E. A. El-Samie, and M. Emam, "Efficient utilization of deep learning for the detection of fabric defects," *Neural Comput Appl*, vol. 36, no. 11, pp. 6037–6050, 2024, doi: 10.1007/s00521-023-09137-0.

[19] G. Revathy and R. Kalaivani, "Fabric defect detection and classification via deep learning-based improved Mask RCNN," *Signal Image Video Process*, vol. 18, no. 3, pp. 2183–2193, 2024, doi: 10.1007/s11760-023-02884-6.

[20] A. International, "Standard Test Methods for Visually Inspecting and Grading Fabrics." [Online]. Available: https://www.astm.org/d5430-93r00.html

[21] J. Jing, D. Zhuo, H. Zhang, Y. Liang, and M. Zheng, "Fabric defect detection using the improved YOLOv3 model," *J Eng Fiber Fabr*, vol. 15, 2020, doi: 10.1177/1558925020908268.

[22] X. Yue, Q. Wang, L. He, Y. Li, and D. Tang, "Research on Tiny Target Detection Technology of Fabric Defects Based on Improved YOLO," *Applied Sciences (Switzerland)*, vol. 12, no. 13, 2022, doi: 10.3390/app12136823.

[23] X. Luo, Q. Ni, R. Tao, and Y. Shi, "A Lightweight Detector Based on Attention Mechanism for Fabric Defect Detection," *IEEE Access*, vol. 11, 2023, doi: 10.1109/ACCESS.2023.3264262.

[24] Q. Liu, C. Wang, Y. Li, M. Gao, and J. Li, "A Fabric Defect Detection Method Based on Deep Learning," *IEEE Access*, vol. 10, 2022, doi: 10.1109/ACCESS.2021.3140118.

[25] H. Jung and J. Rhee, "Application of YOLO and ResNet in Heat Staking Process Inspection," *Sustainability (Switzerland)*, vol. 14, no. 23, 2022, doi: 10.3390/su142315892.

[26] C. Chen, "FF-YOLO: Fashion Fabric Detection Algorithm Based on YOLOv8," *IEEE Access*, vol. 13, pp. 2298–2312, 2025, doi: 10.1109/ACCESS.2024.3524618.

[27] W. Song, D. Lang, J. Zhang, M. Zheng, and X. Li, "Textile Defect Detection Algorithm Based on the Improved YOLOv8," *IEEE Access*, vol. 13, pp. 11217–11231, 2025, doi: 10.1109/ACCESS.2025.3528771.

[28] A. Das Gupta *et al.*, "An approach to automatic fault detection in four-point system for knitted fabric with our benchmark dataset Isl-Knit," vol. 10, Mar. 2024, doi: 10.1016/j.heliyon.2024.e35931.

[29] Y. Matsuzaka and R. Yashiro, "AI-Based Computer Vision Techniques and Expert Systems," 2023. doi: 10.3390/ai4010013.

[30] N. T. Allo, Z. Zainuddin, and others, "A Novel Approach of Hybrid Bounding Box Regression Mechanism to Improve Convergency Rate and Accuracy.," *International Journal of Intelligent Engineering & Systems*, vol. 17, no. 2, 2024.

[31] W. Xiong, Y. Wu, J. Cao, and R. Li, "IB-CBB: an improved spatial index considering intersection based on clipped bounding boxes," *Ann GIS*, vol. 30, no. 2, pp. 233–250, 2024.

[32] M. Sobirov, N. Sharibaev, A. Kayumov, and K. Musayev, "Method of assessment of structural properties of knitted fabrics based on image analysis," in *E3S Web of Conferences*, 2024, p. 3020.

[33] Q. and Z. J. and Y. S. Zhang, "Enhancing YOLOv8 Object Detection with Shape-IoU Loss and Local Convolution for Small Target Recognition," *Informatica*, vol. 49, no. 21, pp. 105–120, 2025.

[34] M. L. Swartz, "Managing digital images," *American Journal of Orthodontics and Dentofacial Orthopedics*, vol. 118, no. 3, pp. 354–358, Sep. 2000, doi: 10.1067/mod.2000.110525.

[35] B. Koonce, "ResNet 50," in *Convolutional Neural Networks with Swift for Tensorflow: Image Recognition and Dataset Categorization*, Berkeley, CA: Apress, 2021, pp. 63–72. doi: 10.1007/978-1-4842-6168-2_6.