information

Article

Fabric Defect Detection in Real World Manufacturing Using Deep Learning

Mariam Nasim [1], Rafia Mumtaz [1,2,*], Muneer Ahmad [3,*] and Arshad Ali [4]

[1] School of Electrical Engineering and Computer Science (SEECS), National University of Sciences and Technology (NUST), H-12, Islamabad 44000, Pakistan; mnasim.msds22seecs@seecs.edu.pk
[2] Center for Computational Science and Mathematical Modelling, Coventry University, Priory Street, Coventry CV1 5FB, UK
[3] Department of Computer Science, University of Roehampton, Roehampton Lane, London SW15 5PH, UK
[4] Software Productivity Strategists, Inc. (SPS), 2400 Research Blvd, Suite 115, Rockville, MD 20850, USA; arshad.ali@spsnet.com
* Correspondence: rafia.mumtaz@seecs.edu.pk (R.M.); muneer.ahmad@roehampton.ac.uk (M.A.)

Abstract: Defect detection is very important for guaranteeing the quality and pricing of fabric. A considerable amount of fabric is discarded as waste because of defects, leading to substantial annual losses. While manual inspection has traditionally been the norm for detection, adopting an automatic defect detection scheme based on a deep learning model offers a timely and efficient solution for assessing fabric quality. In real-time manufacturing scenarios, datasets lack high-quality, precisely positioned images. Moreover, both plain and printed fabrics are being manufactured in industries simultaneously; therefore, a single model should be capable of detecting defects in all kinds of fabric. So training a robust deep learning model that detects defects in fabric datasets generated during production with high accuracy and lower computational costs is required. This study uses an indigenous dataset directly sourced from Chenab Textiles, providing authentic and diverse images representative of actual manufacturing conditions. The dataset is used to train a computationally faster but lighter state-of-the-art network, i.e., YOLOv8. For comparison, YOLOv5 and MobileNetV2-SSD FPN-Lite models are also trained on the same dataset. YOLOv8n achieved the highest performance, with a mAP of 84.8%, precision of 0.818, and recall of 0.839 across seven different defect classes.

Keywords: fabric defect detection; deep learning; YOLOV8; object detection

Citation: Nasim, M.; Mumtaz, R.; Ahmad, M.; Ali, A. Fabric Defect Detection in Real World Manufacturing Using Deep Learning. *Information* **2024**, *15*, 476. https://doi.org/10.3390/info15080476

Academic Editor: Vincenzo Moscato

Received: 11 June 2024
Revised: 27 July 2024
Accepted: 9 August 2024
Published: 11 August 2024

1. Introduction

Fabric is lost as waste due to various defects that occur during manufacturing. This results in monetary losses for textile producers in Pakistan. Mostly, the cause of these defects is machine failure. To check whether the fabric produced is up to the standards of the market, an inspection has to be performed. In developing countries, human labor is involved in this process. A good quantity of skilled workers is required for this manual inspection. The process is physically very time-consuming, and errors can occur due to human fatigue. Therefore, an automated system is required for this task. Recently, along with the development of hardware devices, deep learning-based detection methods have become popular. They are now being applied to different real-world scenarios. One of those scenarios is fabric defect detection.

Although significant research has been conducted for fabric defect detection, it predominantly relies on datasets featuring well-positioned fabric images taken under controlled conditions. However, in real-time manufacturing scenarios, datasets exhibit a variety of challenges. Images are not high-quality and precisely positioned. Moreover, variations in backlight intensity, noise, and blurriness can also significantly impact visual perception. Secondly, the literature on this problem can be segregated into two broad categories, i.e.,

the detection of defects in plain fabrics and the detection of defects in printed fabrics. For printed fabrics, further division can be carried out for regularly printed and irregularly printed fabrics. This is because a huge variety of prints makes it difficult to segregate the pattern from the defect. Therefore, different methodologies have to be applied in all cases. There is very limited exploration of models that can simultaneously detect defects in plain, regularly printed, and irregularly printed fabrics.

Therefore, addressing these limitations, this study hypothesizes that a single object detection model like YOLOv8 can detect all kinds of plain, regularly printed, and irregularly printed fabric defects simultaneously in samples taken in real-world scenarios.

In this study,

- An indigenous dataset taken from Chenab Textile is used. It contains printed and plain fabric images captured in various manufacturing conditions.
- Object detection model YOLOv8, which is computationally faster and requires fewer resources, is trained. For comparison, YOLOv5 and mobilenetv2 SSD FPN lite, which are also deployable on low-resource devices, are trained.
- YOLOv8n produced the best results with a mAP of 84.8%. Followed by YOLOv5n with 84.5% and mobilenetSSD-FPNLite with 77.09%.

The remaining paper is structured as follows: In Section 2, an overview of the existing literature in this field has been provided. Section 3 explains the models along with the dataset. Results have been provided in Section 4, and finally, Section 5 concludes the paper and provides suggestions for further improvement.

2. Related Work

Fabric defect detection is a task that has been focused on a lot in research. Research has been carried out for many years in this field. Algorithms for fabric defect detection can be mainly classified into statistical, spectral, model, and deep learning-based algorithms. Statistical algorithms include methods such as co-occurrence matrix [1] and morphology [2]. They identify the defect by comparing the gray values of the pixels with their surrounding pixels. The detection results for such approaches depend on the size of the window. Moreover, smaller defects are difficult to detect by such methods.

Spectral-based algorithms convert the test image into the frequency domain. Then, for detecting defects, the difference is computed among the spectral coefficients. It includes the Fourier transform method [3], the wavelet transform method [4], and the Gabor transformation [5]. The effectiveness of these methods depends heavily on the choice of filter banks and involves high computation complexities.

Model-based algorithms initially characterize the texture features of non-defective samples using different parameter estimation methods. Subsequently, a defect is detected by determining whether the test image aligns with the established normal texture model. These models, like statistical models, exhibit significant computation complexities and are less effective for detecting small defects. The autoregression model [6] is an example.

Convolutional neural networks have demonstrated exceptional capabilities for representing features in computer vision-related tasks. However, the storage of such large convolutional models and their computational costs are to be simplified. The literature that contains the implementation of various deep learning models for detecting fabric defects can be divided into two main categories: printed fabrics and plain fabrics.

Liu, Zhoufeng, et al. [7] in 2019 used YOLO, a lightweight neural network structure for defecting fabric defects. The basic idea behind this research was to reduce computational costs so that the model could be applied to real-world use cases on embedded devices. Though the accuracy of deep CNNs has exceptionally increased, this has come at the cost of computational expenses and storage provisions. This is a potential barrier to their use in environments with very limited resources, such as mobile phones or other embedded devices. Therefore, the paper proposes 1×1 and 3×3, i.e., small-size convolution layers that reduce dimensionality and fuse features. To increase the model's capability to detect various sizes of defects, multi-scale feature extraction was used. For finding the optimum

size of anchor boxes for YOLO detection, K-means clustering was applied to the fabric defect image dataset. An accuracy of 97.2 was achieved on a fabric image benchmark dataset with 3000 samples and five classes.

Hu, Guanghua, et al. [8] in the same year, 2019 proposed an unsupervised learning approach for the detection of fabric defects, focusing on the challenge of the collection of defective samples for this task. Annotating datasets manually is very time-consuming and expensive. Therefore, this research proposed a deep convolutional generative adversarial network. A standard DCGAN was used with an additional new encoder block, which was used to reconstruct the query image without defects. The output of this encoder, i.e., the reconstructed image and the original image, were subtracted to form a residual map. This map highlights the potential defect regions. A likelihood map is also produced by the model for the image, where every pixel value indicates the chance of defects occurring at that particular point. This likelihood and residual map were used to form a fusion map. The resultant map standardizes gray levels in defect-free regions and displays deviations for defected areas. The model achieved an FNR of 12.09, an accuracy of 51.62, and an FPR of 49.91 on TILDA textile texture and local data samples. However, the drawback was that the model yields noisy segmentation.

Peng, Zhengrui, et al. [9] in 2021 proposed an Attention mechanism and multitask fusion module for fabric defect detection. The attention mechanism makes the networks focus on defects. Whereas multi-task fusion helps the proposed architecture enhance classification using feature concatenation. This fusion module fuses the attention map and classification branches, hence improving classification results, particularly for small-sized defects. The research says that the model is feasible for real-time industrial use cases. The proposed model achieved an F1 score of 0.987, a recall of 0.994, and a precision of 0.98 on the AITEX dataset. However, it has only been implemented for plain fabrics.

Chakraborty et al., in research conducted [10] in the same year, 2021, proposed a deep convolutional neural network to categorize printed fabric defects using fabric images gathered in real-time from industries. Two defect classes that were used are color spots and print mismatches. Simple CNN was explored by experimenting with different hyperparameters, looking for the best ones. The learning rate of 0.0003, batch size 16, and regularization value, λ = 0.001, along with the ReLU activation function, were finalized. The research also applied VGG-16 and VGG-19 architectures to the given dataset. The dataset contained self-collected printed images with 2 classes, color spots, and misprints. VGG16 produced the best results among all three architectures, with a recall of 0.71, precision of 0.70, and accuracy of 72%.

In the same year 2021, Jing et al. [11] proposed TILDA RGBAAM AND IMAGE PYRAMID to detect defects in regular patterned printed fabrics. The first step of this process was to calculate the minimum period for the print of the fabric using RGBAAM. Next, this minimum period was used as a sample template to construct a Gaussian pyramid for the defected image and template image. After that, both the template and the defective image were matched using a similarity measurement method. Lastly, the location of defects in printed fabric was highlighted by using the technique of Laplacian pyramid restoration. The paper states that the proposed model very accurately identifies the periodic unit for print and the location of the defect. However, complex patterns take more time for the model to execute.

Zhang, Jiaqi, et al. [12] in 2022 presented a lightweight MobileNetV2-SSDLite for cloud-edge computing. The model incorporates channel attention and focal loss to address the challenges of detecting small-sized defects and handling the balance between defective and normal samples. Experiments were carried out with 4 different datasets. The proposed approach achieved an accuracy of 84.39 on CF, 93.05 on GF, 71.18 on BPF, and 95.5 on the DRF dataset.

Jia, Zhao, et al. [13] in 2022 implemented improved fasterRCNN for defect detection. This model was presented to solve issues including low accuracy, convergence, and poor results in the detection of tiny defects. The modifications that were made to the faster

RCNN included a ResNet50 backbone instead of VGG16. This solved the problem of the gradient vanishing as the resnet has more depth and residual connections. Another modification was adding an FPN for connecting low- and high-level features to accurately locate and detect small target defects. ROI pooling was used instead of ROI alignment. The benefit was that quantization was canceled, and therefore details were not ignored. Moreover, transfer learning was also applied to decrease training time. The proposed architecture was tested on self-collected yarn samples. Defects including ribbon yarn, broken yarn, holes, stains, etc. were included. The model produced a mAp score of 94.73% on the given dataset. However, the architecture was not deployed in actual production.

Sabeenian, R. S. et al. [14] in 2022 classified 5 different defects in fabrics using the VGG network. The research focused on creating a preprocessing filter for the filtration of nonlinear mixed noise from images and proposed a deep CNN architecture for classifying defects. The proposed work has two stages. In the first step, a pseudo–convolutional neural network, which has been referred to as P-CNN in the paper, is used for image preprocessing. It is a tailored CNN network with three layers resembling traditional convolutional networks. Initial feature extraction layers use weight initialized adaptive window filters. These filter coefficients are initialized using a probabilistic distribution of noise. The PCNN demonstrates outstanding capabilities in rejecting impulse noise in images. Stage 2 comprises a CNN for classifying and detecting defects. An accuracy of 93.92% and a specificity of 92.51 were achieved on the self-collected dataset. However, the model was not able to correctly classify printed fabrics with bands and real-time plain fabric samples with noise.

Liu, Andong, et al. [15] in 2022 proposed a double sparse low-rank decomposition method for detecting defects in printed fabrics with irregular patterns containing high complexities. The proposed model has three sequential stages. Initially, prior information was extracted, consisting of two types of data: the template prior and the defect prior. The template prior was derived from the sparse components, serving as the printing template, while the defect prior was determined by contrasting the defective printed fabric graph with the template fabric graph. Following this, the double sparse low-rank decomposition was carried out to separate the background from the print. Lastly, defect segmentation was carried out by creating the defect mAp by binarizing the saliency mAp of the defects using an optimal threshold segmentation technique. This approach facilitated the clear identification and visualization of the detected defects. The model produces a TPR of 89.29 and an FPR of 0.85 on a self-collected 98 fabric drawings dataset. The model, however, lacks robustness.

Zheng et al. [16] in 2022 proposed an SDANet, which was a siamese FPN for detecting defects in fabrics that are printed. Here, the Siamese feature pyramid network was employed to acquire multi-scale features from the input and standard/template image. An attention module was introduced to detect discrepancies between input and template features. To adjust the positioning error between the standard image and input image features, a self-calibration unit was proposed. Two famous datasets, namely Tianchi Fabric and Tianchi Tile Defect Detection, were used in this research. The model produced a mAp of 47.1 and an accuracy of 83.3%. The main disadvantage of the model was that template images were required for each pattern to detect defects.

Very recently, Li, Long, et al. [17] in 2023 worked on training a robust model on a fabric dataset containing printed and plain fabrics. They implemented a cascade R-CNN on a self-collected dataset with 19 different backgrounds and 9 classes, including stains, holes, wrinkles, and thread ends. To improve the accuracy of the model, certain other techniques were applied. One of them was the "block recognition and detection box merging algorithm" to fully detect defects of small as well as medium size in images with high resolution. For training purposes, large-size, high-resolution images were divided into smaller chunks. Similarly, for inference, large, high-resolution image inputs were segmented into smaller fragments and provided to the model. Later, the detection results of these small fragments were combined to obtain final detection results for the original high-resolution image.

Moreover, a multi-morphology data augmentation method was also proposed and applied. Initial steps involved mean filtering and dynamic thresholding for the extraction of defects by setting the background as either white or black. Then, augmentation techniques like scaling, mirroring, cropping, rotation, morphological processing, etc. were used to alter the shape of defects. They obtained defects through this process, which were randomly merged into fabric images in batches. The results showed mAp of 75.3%; however, only defects present in the patterns available in the dataset were detected effectively.

The summary of the literature review for plain fabrics is presented in Table 1 and that for printed fabrics in Table 2. Following this literature review, two significant research gaps were identified. Firstly, there is a notable segregation in research efforts, with distinct focuses on work related to printed and plain fabrics. Currently, there is an absence of a unified and robust model capable of simultaneously addressing both fabric types with high accuracy. Secondly, the prevailing research emphasis is on datasets that include well-positioned fabric images captured under controlled conditions. However, real-time manufacturing scenarios often lack datasets containing high-quality and precisely positioned images. Additionally, industries concurrently produce both plain and printed fabrics. Therefore, there is a need to develop a robust model capable of accurately detecting defects in fabric datasets generated during production, leveraging recent advancements in deep learning technologies to bridge these identified gaps.

Table 1. Synthesis matrix for plain fabrics literature.

Paper Reference	Dataset	Model Proposed	Research Problem Solved	Accuracy	Limitations
[14]	Self-collected with 5 defect types	Classification using VGG network with preprocessing with Pseudo–Convolutional Neural Network (P CNN)	Filter out nonlinear noise in fabric defect datasets	Accuracy: 93.92, Specificity: 92.51	Cannot classify reality based noisy plain defect samples and banded fabric samples.
[13]	Self collected Yarn samples	Improved faster-RCNN	Poor detection effect for small target defects.	MAp 94.73%	Not deployed in actual production
[12]	CF, GF, BPF, DRF	MobileNetV2-SSDLite based on cloud-edge computing	lightweight CNN for limited resource environments	CF: 84.39 GF: 93.05 BPF: 71.18 DRF: 95.5	Different patterned dataset not considered
[9]	AITEX	Attention mechanism and multi-task fusion module	Enhanced recognition effect especially for Tiny shape defects.	F1score: 0.987, Recall: 0.994, Precision: 0.98	Only implemented for plain fabrics yet
[8]	TILDA textile texture and local	Unsupervised—deep convolutional generative adversarial network (DCGAN)	Collection of defective examples. Manual data annotation is very laborious	FNR: 12.09, Accuracy: 51.62, FPR: 49.91	Yield noisy segmentations.
[7]	Fabric benchmark datasets along with 3000 samples for five classes	YOLO, A lightweight network structure.	High computational cost reduced for systems using embedded devices.	Accuracy: 97.2	Industrial uses still to be explored

Table 2. Synthesis matrix for printed fabrics literature.

Paper Reference	Dataset	Model Proposed	Research Problem Solved	Accuracy	Limitations
[17]	Self-collected printed with 19 different backgrounds	Cascade-RCNN	Printed fabrics defect detection mechanism proposed	mAp: 75.3	Only patterns in the dataset are detected in a good manner
[16]	Tianchi Fabric and Tianchi Tile Defect Detection Datasets	A siamese FPN	Existing techniques depend on extensive annotated datasets and fail on new encountered samples	mAp: 47.1, Accuracy: 83.3	Template images are required
[15]	Self-collected 98 fabric drawings	Double sparse low-rank decomposition method (DSLRD)	Defect detection for irregular print fabrics with complex patterns	TPR: 89.29, FPR: 0.85	Weak robustness
[11]	TILDA	RGBAAM and IMAGE PYRAMID	Identify periodic unit for printed fabric and locate defect	Training time twice less than traditional approaches	Model takes large time for complex patterns and actual production process needs further optimization
[10]	New dataset proposed	Deep convolutional neural network (CNN)	Classify spot and print mismatch	Recall: 0.71, Precision: 0.70, Accuracy: 72	Only 2 types of defects catered

3. Materials and Methods

This section begins with details of the dataset including data collection and preprocessing techniques. Later, the methodology for this study is explained along with details of the model architectures.

3.1. Data Collection and Preprocessing

Currently, fabric defects occurring daily are manually logged in a paper register by workers in the industry. After a manual inspection of the register and looking into the defective sample images collected over five months, some defect classes were shortlisted. This shortlisting was completed based on the frequency of occurrence of that particular class of defect. For certain defects that occur very rarely, the number of data samples was very low. For example, five to six image samples over four to five months. Therefore, such classes were discarded. Seven classes, including baekra, color issues, contamination, cut, gray stitch, selvet, and stain, are included in this study. Color issues include color spots and discoloration defects. Examples of data samples for each class are shown in Figure 1. It can be seen that the dataset includes samples of plain fabric, regularly printed fabric, and irregularly printed fabric. Details of the defects, as provided by the industry, are presented in Table 3.

The dataset was collected by the workers in the factory. The raw dataset was shared using a drive link. Originally, the folders were sorted according to the dates, inside which subfolders for different defect classes were present. During the data preprocessing phase, individual folders were created for each class, and images from different dates were manually added to the corresponding folders.

The dataset was then annotated using roboflow. As for most detection models, including YOLO, we need square bounding boxes; therefore, square bounding boxes were manually formed for all the defective samples. The dataset had a class imbalance problem. Samples for certain classes, like stain, were large as compared to other classes. To solve that, firstly, for classes that contained fewer samples, more augmentation was performed for them. Augmentation techniques, including image rotation and flipping, were mainly applied.

Figure 1. Some images from Chenab dataset.

Table 3. Detailed description of defects.

Defect Type	Description
Contamination	During weaving, if another thread comes into contact with the original thread of the cloth, it leaves a visible mark on the cloth in the shape of a line. This defect is known as contamination.
Selvet	When rolling, if one layer of cloth folds slightly, the pressure from other layers keeps the fold in place, leaving a folded mark. This defect is known as selvet.
Gray Stitch	When one piece of cloth ends, or cloth is torn by the machine while connecting it to another cloth, we use a stitch known as a joint connection, also referred to as a gray stitch.
Cut	If there is any cut in the cloth caused by machines or any other reason, it falls under this category. Cuts can also appear on the edges of clothes.
Baekra	When the printing machine stops, it sometimes leaves marks in lines or causes major outages, destroying the design and resulting in no pattern. The defect is known as baekra.
Color Issues	1. Incomplete color coverage in print, results in areas where the color is lighter. 2. Printing defects that appear as spots if color or any debris comes into contact with the cloth during printing. 3. When one color mixes into another, it is called color mixing.
Stains	Can be of various types, such as oil, dust, or rust stains. They can appear as spots or be spread out on the cloth.

The dataset was split into the train, test, and valid folders and finally exported in YOLOv5, YOLOv8, and tfrecord format from roboflow.

After initial training of the model, results showed that for certain classes, the mAp values were not satisfactory. Therefore, for these classes, image samples from some publicly

available datasets present on roboflow were added to the dataset, and the model was retrained. Images from thesis dataset [18], FabricDefectDet2 [19], and defect_1 [20] were chosen and incorporated into the dataset. This choice was made because the samples in these datasets exhibited properties similar to ours; most images were unflattened and had varying orientations.

The final dataset contained around 2800 samples. Statistics showing the number of samples for different classes in the dataset are shown in Table 4.

Table 4. Annotated data samples for each class.

Classes	Train	Valid	Test
baekra	175	48	17
color issues	86	15	10
contamination	148	53	14
cut	272	75	41
gray stitch	228	63	33
selvet	218	64	30
stain	662	208	112

Dataset Innovation and Characteristics

The datasets mentioned in the literature section, e.g., TILDA [21], TIANCHI [22,23], AITEX [24], all are extensive, contributing effectively to defect identification. However, they often include images that are well-positioned and of high quality, focusing solely on the defect area without background distractions. Our dataset captures images directly from the production line without any manual positioning or enhancement. This "as-produced" approach ensures the addition of natural imperfections and background elements encountered in the manufacturing process. Therefore, our dataset offers a more realistic training and prediction environment for defect detection models.

The main characteristics of our dataset include:

- **As-Produced Variability:** Unlike many existing datasets, our dataset comprises images captured directly from an operational textile production environment at Chenab Textile. The fabrics are captured as they are produced, unflattened, and unaltered, which simulates a more realistic detection scenario.
- **Variability in Orientation:** Fabrics are captured at various angles and orientations, simulating real production line conditions.
- **Noise and Imperfections:** Images include real-world imperfections such as noise and blurriness.
- **High Diversity of Defects:** The dataset features a wide range of defect types, including cuts, stains, gray stitches, and more, ensuring comprehensive coverage of potential issues.
- **Background Elements:** Unlike datasets that focus solely on the defect area, our images include background elements that are present in the production environment. This adds to the complexity of the images and simulates the challenges faced in real-world fabric defect detection.
- **Mixed Fabric Types:** Incorporating plain, regularly printed, and irregularly printed fabrics ensures that models trained on our dataset can generalize well to different fabric patterns and types.

Hence, we provide a resource that bridges the gap between controlled experimental datasets and the variability seen in industrial settings. This makes our dataset particularly valuable for developing robust fabric defect detection systems applicable in real-world scenarios. The dataset can be accessed at the following link: https://tinyurl.com/488uhkhy. (accessed on 26 July 2024).

3.2. Methodology

A generalized overview of methodology is shown in Figure 2. The dataset collected from Chenab textiles was augmented and annotated as mentioned in Section 3.1. Then deep learning models were selected for this particular problem. This use case demanded architectures that are computationally faster but can be deployed in less resource environments. This is because the system is to be deployed in high-speed fabric production environments where cameras are strategically positioned to scan continuously produced fabric, ensuring the prompt detection of defects. After a detailed literature review, it was seen that YOLO and mobilenetSSD FPN lite are the object detection algorithms suitable for such scenarios. In MobilenetSSD FPN Lite, mobilenetv2 serves as the backbone. It is a lightweight convolutional neural network that employs depthwise separable convolutions and inverted residual blocks. These features make it suitable for real-time applications with limited computational resources. As far as YOLO is concerned, it frames object detection as a single regression problem. YOLO processes the entire image in one forward pass through the network, unlike other models that process regions of interest separately, which is more time-consuming. This approach reduces the computational overhead and latency.

Therefore, YOLOv8 is trained to detect defects. For comparative purposes, SSD-mobilenetv2-FPNLite and YOLOv5 are also applied to the same dataset to get results. The detailed architectures of these models have been explained in later sections. After training the models, predictions were made for test data samples. Based on the mAPs, classes showing lower accuracies were enhanced by incorporating images from publicly available datasets, resulting in improved outcomes.

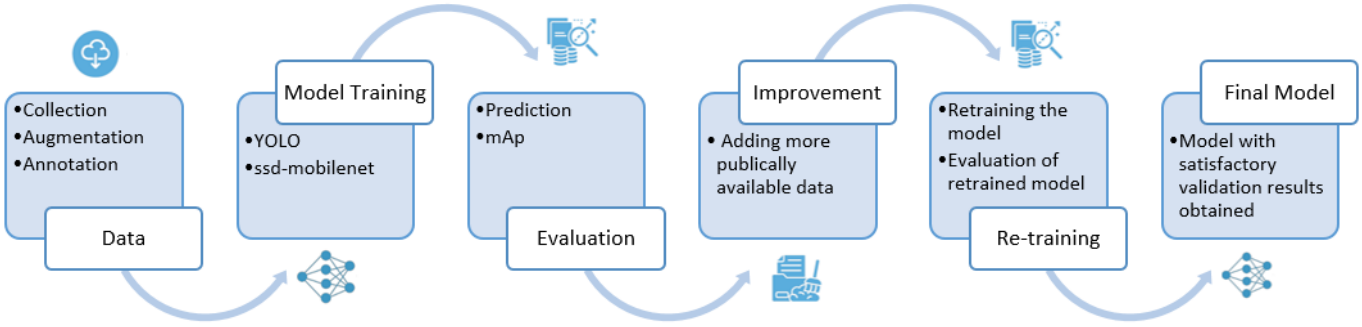

Figure 2. Overview of proposed methodology.

3.2.1. MobileNetV2 SSD FPN-Lite

MobileNetV2 SSD FPN-Lite is an architecture tailored for object detection tasks, particularly optimized for deployment on mobile and embedded devices. It combines the lightweight MobileNetV2 backbone with the multi-scale feature representation of FPN to enable efficient and accurate object detection on mobile and embedded platforms. Here is an overview of its architectural details:

Backbone Network (MobileNetV2): MobileNet was introduced by Howard et al. in 2017 [25], and in 2018, Sandler et al. introduced MobileNet V2 [26]. MobileNet V2 demonstrated superior accuracy relative to MobileNet V1, despite utilizing fewer parameters. In this particular model, MobileNetV2 serves as the backbone network for feature extraction. MobileNetV2 employs depthwise separable convolutions and inverted residual blocks to balance model size, speed, and accuracy. These features make it suitable for real-time applications with limited computational resources.

Feature Pyramid Network (FPN): FPN is integrated into the architecture to address scale variation and enhance feature representation. FPN generates a hierarchical set of feature maps at different spatial resolutions, allowing the model to detect objects at various scales. This multi-scale representation is crucial for accurately detecting objects of different sizes in the input image. Hence, different-sized fabric defects can be accurately detected.

SSD Head: The SSD head is responsible for predicting object bounding boxes and class probabilities. It consists of a series of convolutional layers, followed by prediction layers. These prediction layers generate class scores and bounding box offsets for predefined anchor boxes across different feature maps. The SSD head enables efficient object localization and classification at multiple scales.

Lite Optimization: The "Lite" version of MobileNetV2 SSD FPN incorporates optimizations to reduce computational complexity and memory usage while maintaining reasonable detection performance. These optimizations may include techniques like channel pruning, quantization, and architectural modifications tailored for resource-constrained environments. By leveraging these optimizations, MobileNetV2 SSD FPN-Lite achieves a good balance between detection accuracy and efficiency, making it suitable for deployment on devices with limited hardware resources.

3.2.2. YOLOv5

Among various object detection algorithms, the YOLO framework [27] has distinguished itself for achieving an impressive equilibrium between speed and accuracy.

The YOLOv5 architecture represents a significant advancement in the YOLO (You Only Look Once) series, renowned for its real-time object detection capabilities. As mentioned in [28], YOLOv5 was released in 2020, a few months after YOLOv4. YOLOv5 is designed with a streamlined and efficient architecture tailored for high-speed inference while maintaining competitive accuracy. Here is a detailed overview of its architecture:

Backbone Network: YOLOv5 utilizes a modified version of the CSPDarknet53 backbone network, which is derived from the Darknet architecture. CSPDarknet53 incorporates cross-stage partial connections (CSP) to facilitate efficient information flow across network stages. These connections enhance feature representation and contribute to the model's ability to capture complex patterns in images effectively.

Feature Pyramid Network (FPN): YOLOv5 incorporates a path aggregation network, an extension of a feature pyramid network (FPN). A FPN enables multi-scale feature extraction by combining features from different network layers. It achieves this by introducing lateral connections that fuse low-level and high-level features, resulting in feature maps with rich spatial information at multiple scales. This capability is crucial for detecting objects of various sizes and scales in images.

Detection Head: The detection head of YOLOv5 processes the multi-scale features extracted by the backbone and FPN networks to generate bounding box predictions, confidence scores, and class probabilities. It consists of a series of convolutional layers followed by detection-specific operations, such as anchor box assignment and non-maximum suppression (NMS). YOLOv5 predicts bounding boxes using anchor boxes, which are predefined boxes of different aspect ratios and scales, to accurately localize objects in images.

Training and Inference: YOLOv5 is typically trained using large-scale labeled datasets, such as COCO (common objects in context) or VOC (Visual Object Classes), with techniques like stochastic gradient descent (SGD) with momentum and weight decay. The model is optimized using loss functions tailored for object detection tasks, including binary cross-entropy for objectness prediction and mean squared error for bounding box regression. Additionally, techniques like focal loss may be employed to address class imbalances and improve training stability. Once trained, YOLOv5 is capable of performing real-time object detection on images or videos with remarkable speed and accuracy. The model can be deployed on various platforms, including CPUs, GPUs, and specialized hardware accelerators, making it suitable for a wide range of applications, such as autonomous vehicles, surveillance systems, and robotics.

Ultralytics offers five versions of the model: nano, small, medium, large, and extra-large, each with a different convolution module width and depth to cater to specific use cases and hardware requirements.

3.2.3. Yolov8

Ultralytics, the creators of YOLOv5, introduced YOLOv8 [29] in January 2023. The YOLOv8 architecture represents an evolution of the YOLO series, introducing several enhancements and modifications compared to YOLOv5. While maintaining the core principles of real-time object detection and high accuracy, YOLOv8 incorporates unique features and improvements. Architectural details, as mentioned in [28], are provided in the following paragraphs.

Backbone Network: YOLOv8 adopts a backbone similar to YOLOv5, incorporating alterations in the CSPLayer, now known as the C2f module. This C2f module (cross-stage partial bottleneck with two convolutions) merges high-level features with contextual information, enhancing detection accuracy. Employing an anchor-free model and a decoupled head, YOLOv8 processes objectness, classification, and regression tasks independently, allowing each branch to focus on its specific task, thus refining the overall model accuracy.

Output Layer: In the output layer, YOLOv8 applies the sigmoid function as the activation for objectness scores, indicating the likelihood of an object within the bounding box, while the class probabilities utilize the softmax function, representing object probabilities across potential classes.

Segmentation Model: Additionally, YOLOv8 introduces a semantic segmentation model named YOLOv8-Seg, employing a CSPDarknet53 feature extractor as the backbone, followed by a C2f module. YOLOv8-Seg showcases state-of-the-art results in various object detection and semantic segmentation benchmarks, maintaining high speed and efficiency. For loss functions, YOLOv8 utilizes DFL and CIoU for bounding-box loss and binary cross-entropy for classification. This enhances performance, especially for smaller objects.

Hence, YOLOv8, having an improved backbone network and an anchor-free architecture with multi-scale prediction capabilities, outperforms previous versions in terms of accuracy and speed. Ref. [29] presents a detailed architecture of YOLOv8.

Ultralytics offers five versions for this model as well: nano, small, medium, large, and extra-large, each with a different convolution module width and depth to cater to specific use cases and hardware requirements.

3.3. Training

Training and inference were conducted using pytorch with the Ultralytics framework. The dataset exported from roboflow in yolov8 format contains a "data.yaml" file with details such as the total number of classes and paths to train, test, and validate directories. The YOLOv8 package was installed using Ultralytics, and a "data.yaml" file was provided to the model as a dataset for training with an image size set to 640 and the number of epochs set to 500. Two variants of yolov8, i.e., nano and small, were trained. The final mAP values were approximately the same for both variants; therefore, yolov8n was chosen to be the final model, as it constitutes a smaller number of parameters as compared to yolov8s.

Different hyperparameter combinations were used to train different YOLOv8n models. The default hyperparameter values for YOLOv8n with image size 640, epochs 500, and batch size 16 were proven to produce the best results. This included an initial learning rate and a final learning rate both set to 0.01, a momentum of 0.937, and a weight decay of 0.0005. A warm-up phase of 3.0 epochs was implemented to stabilize the training. Other critical parameters, such as box and class loss weights tuned to 7.5 and 0.5 respectively, were used. For the rest of the YOLO specific parameters, hsv_h = 0.015, hsv_s = 0.7, hsv_v = 0.4, degrees = 0.0, translate = 0.1, scale = 0.5, shear = 0.0, perspective = 0.0, flipud = 0.0, fliplr = 0.5, mosaic = 1.0, mixup = 0.0, copy_paste = 0.0 were taken. The albumentations library is also integrated in YOLOv8 by Ultralytics, applying occasional blur, grayscale conversion, and contrast limited adaptive histogram equalization to augment the dataset. The optimizer employed is stochastic gradient descent with a learning rate of 0.01 and momentum of 0.9. Training graphs produced during the training of this final model are shown in Figure 3. The graphs show that with the increasing number of epochs, different losses drop and mAp

values increase, indicating an increase in accuracies. The results produced by this model are explained in Section 4.

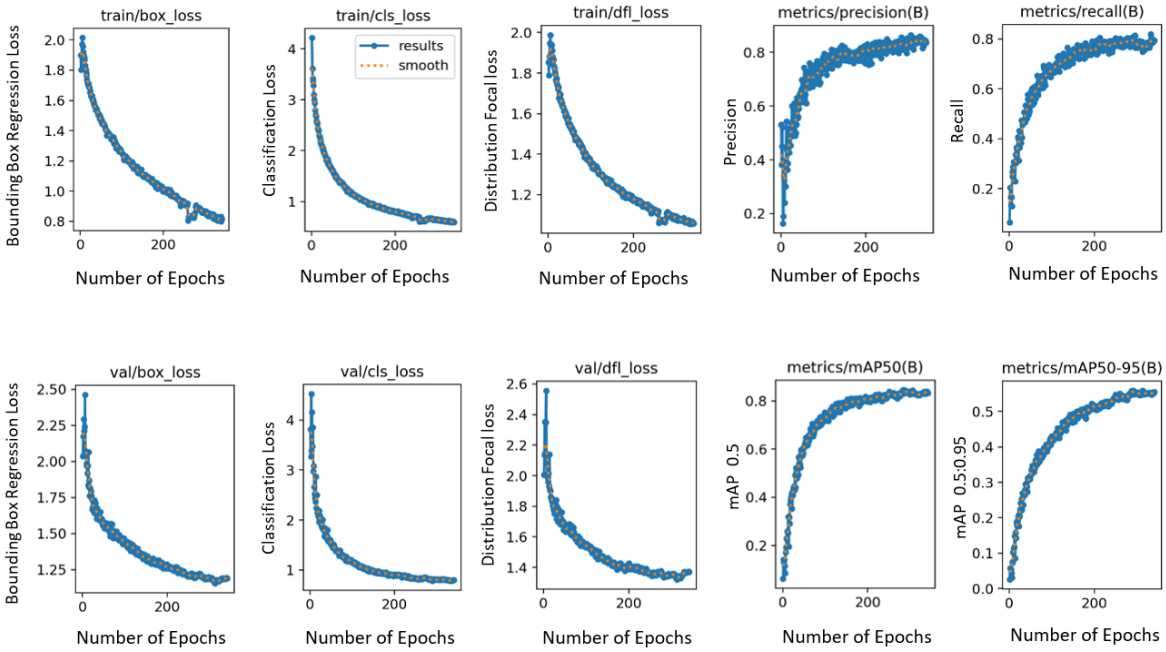

Figure 3. Graphs produced during training of YOLOv8.

Keeping learning rate and optimization the same, but changing YOLOv8 parameters (box = 0.1 cls = 0.4 hsv_h = 0.01 hsv_s = 0.8 hsv_v = 0.5 degrees = 10.0 translate = 0.2 scale = 0.7 shear = 0.1 perspective = 0.05 flipud = 0.05 fliplr = 1.0 mosaic = 1.0 mixup = 0.3 copy_paste = 0.2) with image size set to 416, drastically dropped the accuracies(mAp). Whereas keeping YOLOv8 parameters same but changing learning rate and optimization values (lr0 = 0.005 lrf = 0.02 momentum = 0.95 weight_decay = 0.0001 warmup_epochs = 1.0 warmup_momentum = 0.9 warmup_bias_lr = 0.2) with image size set to 416, produced good results, but they were still low as compared to the results obtained by default parameters. Therefore, default hyperparameters were selected as final parameters, and later they were used to train YOLOv5 as well for comparison purposes.

Similar was the case with yolov5. It was cloned using ultralytics. A "data.yaml" file exported in YOLOv5 format from roboflow was provided to the model for training. YOLOv5n and YOLOv5s were trained for 500 epochs and an image size of 640 with default parameters to produce approximately similar results; hence, YOLOv8n was chosen as the final model. The graphs produced during training of YOLOv5 indicating different losses are displayed in Figure 4. Results produced by this model are explained in Section 4.

The ssd-mobilenet-v2-fpnlite model can be implemented by cloning it through [30] and applying it to the dataset exported in tfrecord format using tensorflow. The model was trained twice. For the first experiment with 6000 steps and batch size 16, a momentum optimizer with learning_rate_base: 0.01 and warmup_learning_rate: 0.0026666 with 1000 warmup_steps was used. It produced an mAP@0.5 of 62.71% and mAP@0.5:0.95 of 27.5%. Another model trained for 20,000 steps with batch size 16 used momentum optimizer with learning_rate_base: 0.08 and warmup_learning_rate: 0.026666 with 1000 warmup_steps. It produced an mAP@0.5 of 77.09% and mAP@0.5:0.95 of 39.61%. Results for this second model have been presented in detail in Section 4. The loss graphs produced during the training of this model are displayed in Figure 5. The graph shows that with the increasing number of steps, the classification loss and localization loss decrease, finally contributing towards decreasing the overall loss.

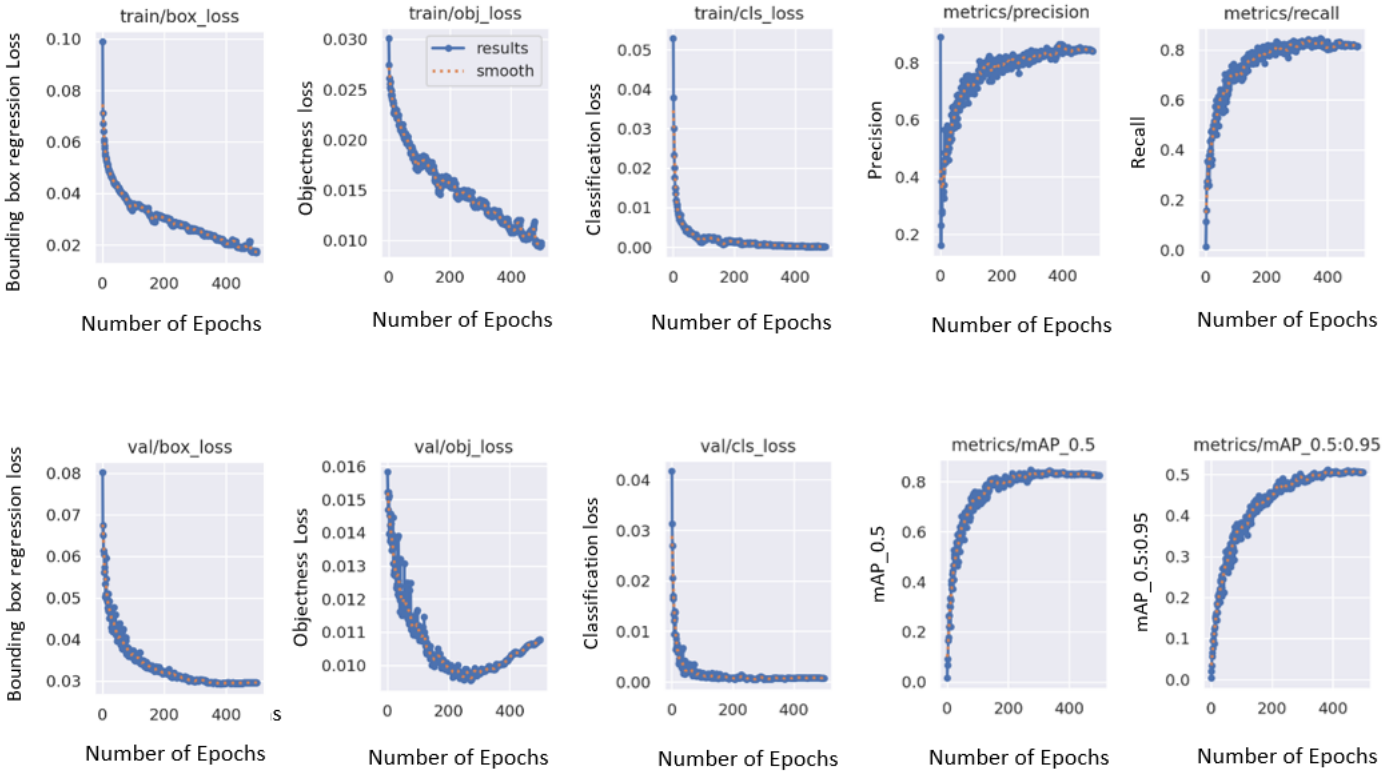

Figure 4. Graphs produced during training of YOLOv5.

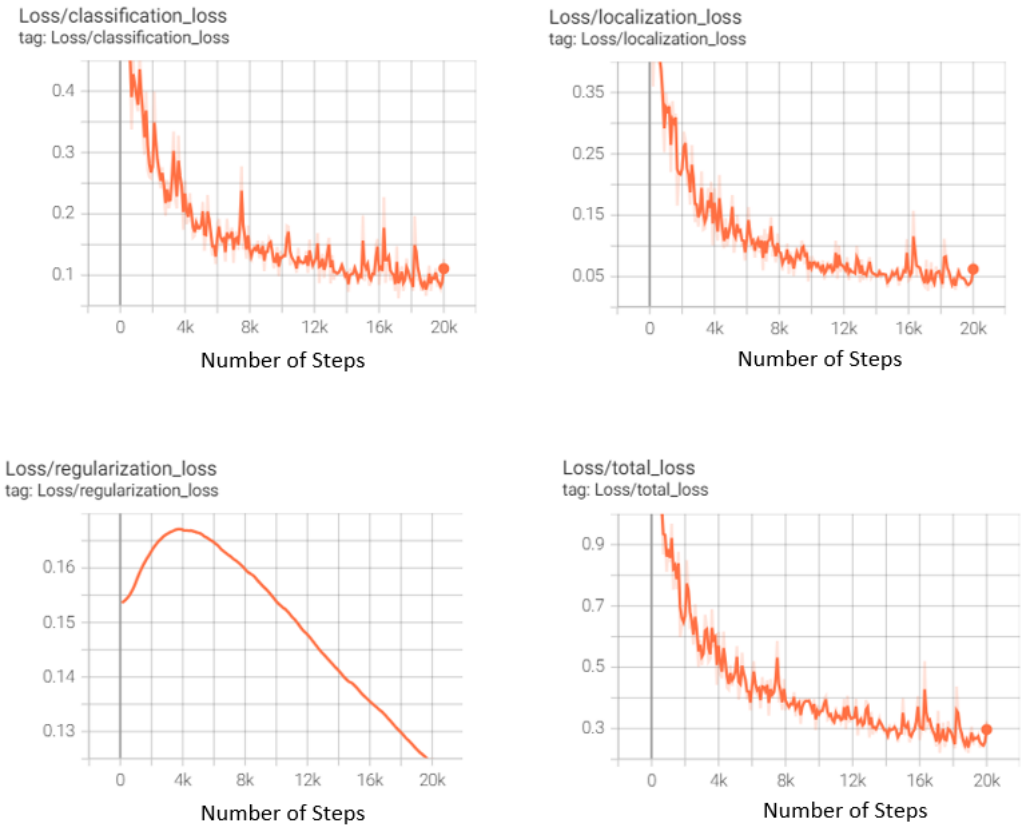

Figure 5. Loss graphs produced during training of the ssdmobilenet-fpnlite.

Recently, YOLOv10 was released and subsequently we trained it. We used the final hyperparameter values selected for YOLOv8 and YOLOv5 to enable a direct comparison. The trained model yielded a mAP50 of 82.6 and a mAP50-95 of 56.7 on the test dataset.

4. Results

Mean average precision, mostly written as mAp is used mainly as an evaluation metric in this study. However, along with mAp, recall(R) and precision(P) have also been displayed for YOLOv5 and YOLOv8.

Outcomes generated by MobileNetSSD indicate a comprehensive mean average precision (mAP) of up to 77.09% on the test set. Detailed results are presented in Table 5. It can be seen from the table that mAp for contamination class is greatest, as the model learns patterns for detecting it in the best manner. This is followed by baekra with 91.63% and stain with 90.07%. Precision and recall values can also be viewed in the table. Precision is highest for stain with a value of 0.671. Recall is highest for contamination with a value of 1.0.

Table 5. Results for MobilenetSSD.

Classes	mAP@0.5	mAP@0.5–0.95	Precision	Recall
all	77.09%	39.61%	0.525	0.881
baekra	91.63%	61.82%	0.411	0.958
colorissues	56.85%	27.32%	0.429	0.714
contamination	92.43%	30.83%	0.515	1.000
cut	63.92%	34.35%	0.530	0.745
graystitch	70.18%	38.65%	0.424	0.875
selvet	74.57%	43.07%	0.432	0.879
stain	90.07%	41.20%	0.671	0.922

Table 6 provides details of results generated by YOLOv5 on training and test sets, respectively. For training sets, the overall mAp is 83.3%. The individual class accuracy shows remarkable results for contamination class with mAp of 99.5%, stain with mAp 92.2%, and cut with 87.3%. For the test set, the overall mAp is 84.5% with exceptional results for contamination with mAp 99.5%, baekra and selvet with mAp 91.2%, and stain with mAp 93.6%. The remaining values can be viewed in the table. It also lists precision and recall values for all classes.

Table 6. Results for training and test datasets for YOLOv5.

	Training Accuracies				Test Accuracies			
Classes	**mAP@0.5**	**mAP@0.5–0.95**	**Recall**	**Precision**	**mAP@0.5**	**mAP@0.5–0.95**	**Recall**	**Precision**
all	83.3%	51.1%	0.813	0.855	84.5%	52.5%	0.845	0.827
baekra	84.1%	59.2%	0.797	0.833	91.2%	58.6%	0.87	0.874
colorissues	65%	34.6%	0.7	0.8	72.2%	35.5%	0.798	0.677
contamination	99.5%	56.8%	1	0.992	99.5%	61.6%	1	0.968
cut	87.3%	55.4%	0.822	0.903	71.2%	46.9%	0.766	0.698
graystitch	74.2%	45.2%	0.667	0.746	72.9%	49.3%	0.667	0.778
selvet	81%	44.4%	0.793	0.801	91.2%	60.3%	0.867	0.893
stain	92.2%	62.4%	0.913	0.911	93.6%	55.2%	0.948	0.897

Table 7 provides details of results generated by YOLOv8 on training and test sets, respectively. For training sets, the overall mAp is 83.8%. The individual class accuracy shows remarkable results for contamination class with mAp of 99.5%, stain with mAp 91.3%, and baekra with 89%. For the test set, the overall mAp is 84.8% with exceptional results for contamination with mAp 95.7%, baekra with mAp 94.5%, and stain with mAp 92.2%. The remaining values can be viewed in the table. It also lists precision and recall values for all classes.

Table 7. Results for training and test datasets for YOLOv8.

	Training Accuracies				Test Accuracies			
Classes	**mAP@0.5**	**mAP@0.5–0.95**	**Recall**	**Precision**	**mAP@0.5**	**mAP@0.5–0.95**	**Recall**	**Precision**
all	83.8%	55.9%	0.797	0.825	84.8%	57.5%	0.839	0.818
baekra	89%	68.4%	0.835	0.788	94.5%	72.5%	0.989	0.922
colorissues	68.2%	39.3 %	0.654	0.71	75.9%	39%	0.81	0.732
contamination	99.5%	61.4%	1	0.996	95.7%	63%	0.941	0.866
cut	84.5%	57.1%	0.81	0.892	70.2%	48.6%	0.696	0.671
graystitch	72%	52.3%	0.609	0.713	74.7%	55.4%	0.683	0.785
selvet	82%	50.3%	0.785	0.776	90.5%	65%	0.828	0.87
stain	91.3%	62.4%	0.883	0.902	92.2%	58.9%	0.928	0.879

Detection Speed

On 280 test data samples at shapes (32, 3, 640, 640), the detection speed for yolov8 was around 11.5 ms per image. This includes 1.4 ms for preprocessing, 8.4 ms for inference, and 1.7 ms for postprocessing. For yolov5, it was around 15.9 ms, with 0.2 ms for pre-processing, 7.8 ms for inference, and 7.9 ms for postprocessing steps.

5. Discussion

To visualize the comparison of mAp values produced overall and for individual classes, a column chart has been shown in Figure 6.

It can be seen that YOLOv8 produced better results with an overall mAP of 84.8%. If we look into individual class mAP values, then results for ssd-mobilenet are low compared to YOLO models. However, YOLOv5 and YOLOv8 produced approximately similar results, with one exceeding the other in certain classes. For instance, in cases of contamination, cut, selvet, and stains, YOLOv5 performed a certain percentage better than YOLOv8. However, for baekra, color issues, and gray stitch, YOLOv8 outperformed YOLOv5 with good percentages, producing the best results overall.

As for certain classes, YOLOv8 outperformed YOLOv5. While for others, YOLOv5 outperformed YOLOv8. Therefore, to improve accuracy, we tried using a class wise based ensemble learning method. This technique improved accuracies for certain classes like baekra, color issues, and contamination. However, for the other 4 classes, the mAp dropped, dropping overall accuracy to 82%. This outcome highlights the complex nature of ensemble methods, where performance gains in some areas may not always translate to overall improvements.

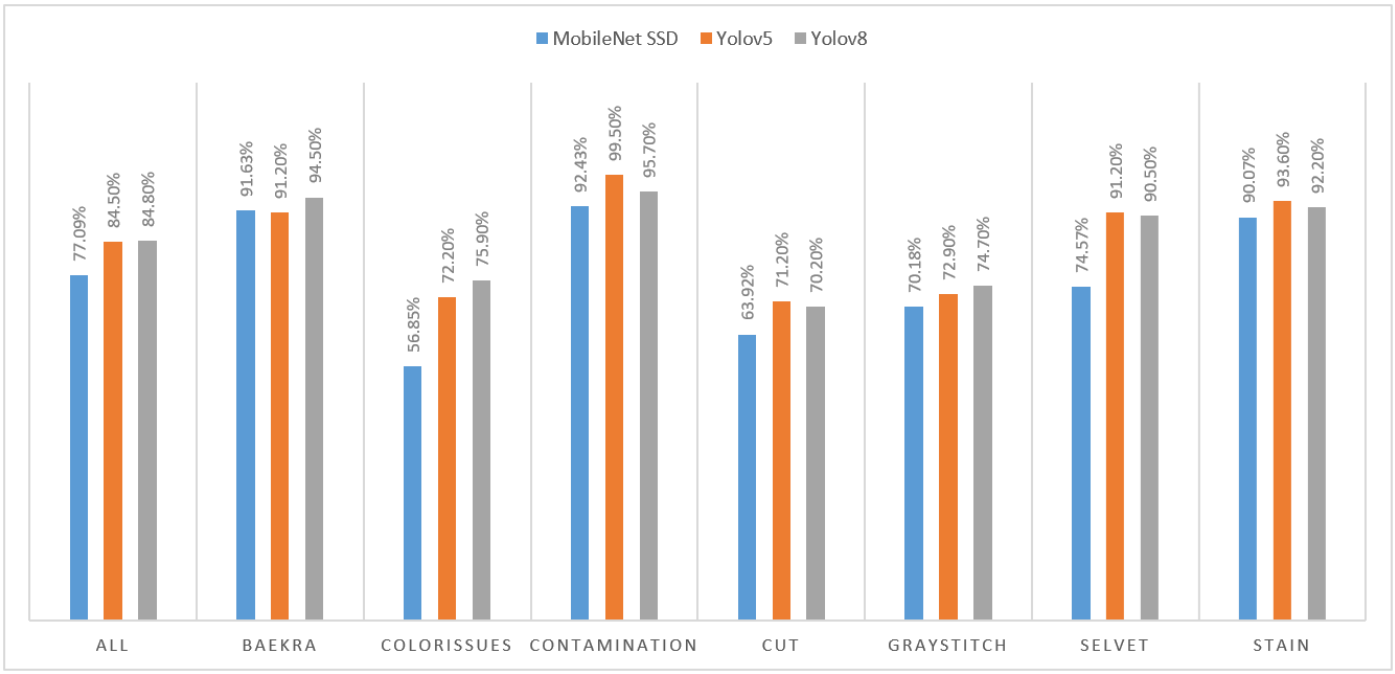

Figure 6. Comparison of results produced by different models.

YOLOv10, though being latest produced mAp50 of 82.6 and mAp50–95 of 56.7 on the test dataset. One of the reasons could be that the selected final hyperparameters may not be optimal for this model in this specific use case. Therefore, to determine whether YOLOv10 outperforms YOLOv8 for this task, in the future, we can explore all the hyperparameter combinations for YOLOv10 to ascertain whether they are optimal or if different settings might yield better results.

Some of the predictions made by YOLOv8 on test images are shown in Figure 7. At least one test sample for each class has been shown. It can be seen that defects have been accurately detected, with bounding boxes covering the defect accurately and with good confidence scores. The confidence score for the cut class in the shown test sample is very high with a value of 0.90. Stains have also been accurately detected with a confidence value of 0.87 for printed fabric and 0.80 for the plain fabric sample. The baekra defect has also been detected with 0.86 confidence, followed by gray stitch with 0.85 and contamination with 0.81. For color issues, the confidence score is low, with a value of 0.49.

Therefore, the latest state-of-the-art object detection model, i.e., YOLOv8, can detect defects in plain and printed fabrics (with regular and irregular prints) simultaneously. As we can see from the results, certain majorly occurring defects can be detected well. However, the mAp for class "color issues" is quite low. To improve this, one of the approaches could be to include more data samples for this particular class so that patterns for that class are learned well. In addition, the current study encompasses a small number of broad categories of defects commonly detected in the textile sector of Pakistan, and it solely depends upon the data provided by a specific manufacturer. More variations can be added to the dataset, which is useful for improvement.

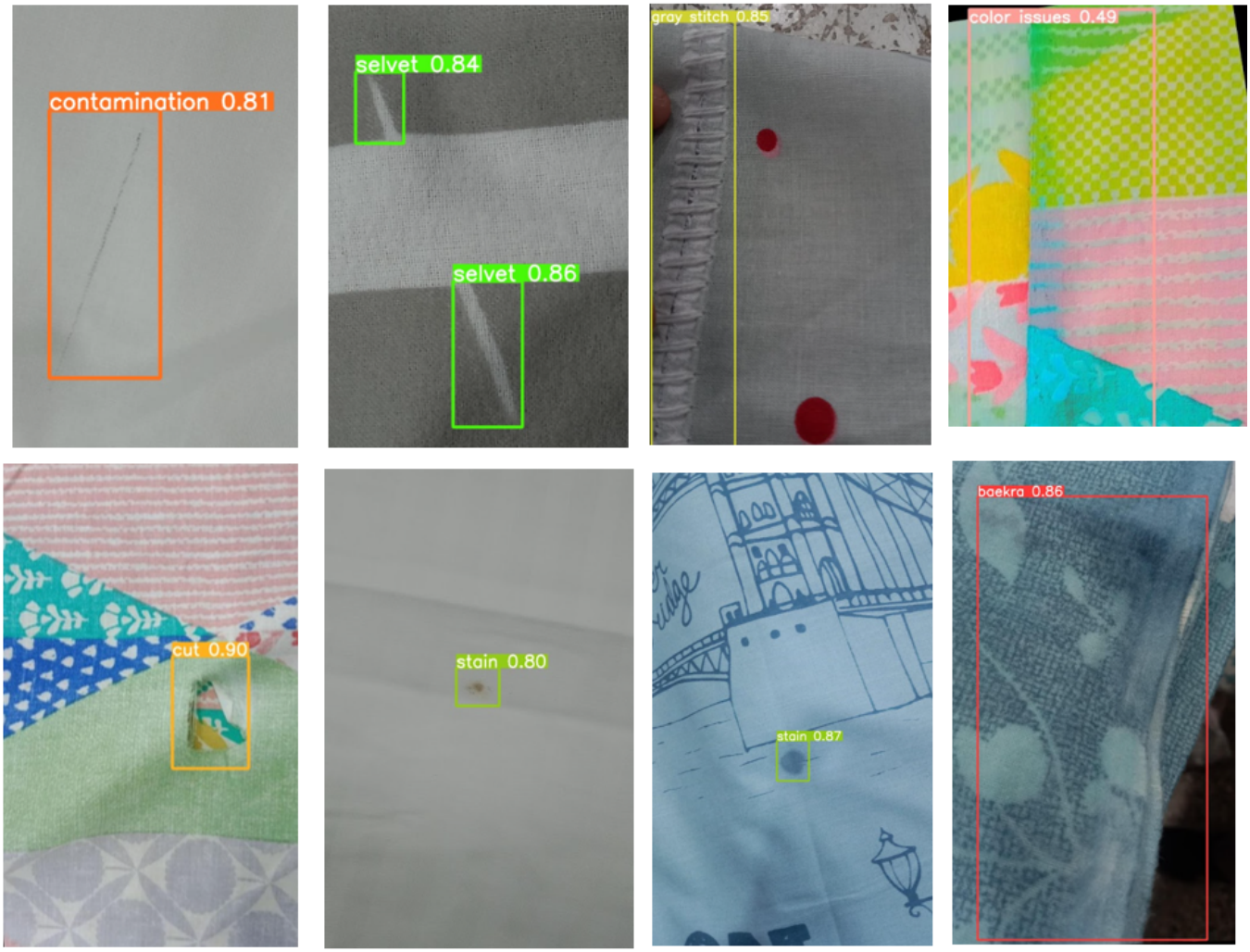

Figure 7. Some results of YOLOv8 trained model on test images.

6. Conclusions and Future Work

Our approach involves training and testing an object detection model, YOLOv8, on the Chenab Textile dataset. YOLO has been chosen as it is computationally faster, requires fewer resources, and can be easily deployed in real time on low-resource hardware devices for detecting defects. On samples provided for seven defect classes, including stains, cut, contamination, baekra, gray stitch, color issues, and selvet, YOLOv8 produced a mAp of about 84.8%. In comparison, YOLOv5 achieved 84.5% mAP, while MobilenetSSD FPNLite attained mAP of 77.09% on the same dataset.

The subsequent pivotal phase in our work will focus on real-time testing of this model in high-speed fabric production environments. To integrate this trained model into existing manufacturing workflows, high-resolution cameras connected via high-speed Ethernet cables will be installed above the rolling sheets, with bright LED lighting to ensure optimal image quality. On the software side, the system will run on Ubuntu or Windows with the deep learning framework PyTorch and Ultralytics installed to set up the environment. Trained YOLOv8 model weights will be loaded and configured for real-time inference. A database (e.g., MySQL or PostgreSQL) will also be required for storing detection results. Additionally, a user interface can be developed using visualization libraries like Matplotlib to display real-time detection results and defect data analysis, ensuring the smooth operation of the system.

Author Contributions: Conceptualization, M.N., R.M., M.A., and A.A.; methodology, M.N., R.M., M.A., and A.A.; software, M.N., R.M., and A.A.; validation, M.N., R.M., and M.A.; investigation, M.N., R.M., M.A., and A.A.; data curation, M.N., and R.M.; writing—original draft preparation, M.N., R.M., M.A., and A.A.; writing—review and editing, M.N., R.M., M.A., and A.A.; visualization, M.N., R.M., M.A., and A.A.; supervision, R.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data can be made available by requesting the authors through email.

Acknowledgments: The research work is conducted in the NUST-Coventry Internet of Things Lab (NCIL) at NUST-SEECS, Islamabad, Pakistan. We thank Hashmat Malik, CEO of SPS (Software Productivity Strategists, Rockville USA) for his unwavering support and invaluable assistance throughout this research. The contributions of Malik and the entire SPS team, including their provision of data and initial support, were paramount to the successful completion of this study.

Conflicts of Interest: Arshad Ali is employee of Software Productivity Strategists, Inc. (SPS) company. The authors declare that they have no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

CIoU	Complete Intersection Over Union
dfl_loss	Distribution Focal Loss
cls_loss	Classification Loss
obj_loss	Objectness Loss
box_loss	Bounding Box Regression Loss
ms	millisecond

References

1. Zhu, D.; Pan, R.; Gao, W.; Zhang, J. Yarn-dyed fabric defect detection based on autocorrelation function and GLCM. *Autex Res. J.* **2015**, *15*, 226–232. [CrossRef]
2. Mak, K.L.; Peng, P.; Yiu, K.F.C. Fabric defect detection using morphological filters. *Image Vis. Comput.* **2009**, *27*, 1585–1592. [CrossRef]
3. Hu, G.; Wang, Q.; Zhang, G. Unsupervised defect detection in textiles based on Fourier analysis and wavelet shrinkage. *Appl. Opt.* **2015**, *54*, 2963. [CrossRef] [PubMed]
4. Zhu, Q.; Wu, M.; Li, J.; Deng, D. Fabric defect detection via small scale over-complete basis set. *Text. Res. J.* **2014**, *84*, 1634–1649. [CrossRef]
5. Jia, L.; Chen, C.; Liang, J.; Hou, Z. Fabric defect inspection based on lattice segmentation and Gabor filtering. *Neurocomputing* **2017**, *238*, 84–102. [CrossRef]
6. Cohen, F.S.; Fan, Z.; Attali, S. Automated inspection of textile fabrics using textural models. *IEEE Trans. Pattern Anal. Mach. Intell.* **1991**, *13*, 803–808. [CrossRef]
7. Liu, Z.; Cui, J.; Li, C.; Wei, M.; Yang, Y. Fabric defect detection based on lightweight neural network. In Proceedings of the Chinese Conference on Pattern Recognition and Computer Vision (PRCV), Xi'an, China, 8–11 November 2019; pp. 528–539.
8. Hu, G.; Huang, J.; Wang, Q.; Li, J.; Xu, Z.; Huang, X. Unsupervised fabric defect detection based on a deep convolutional generative adversarial network. *Text. Res. J.* **2020**, *90*, 247–270. [CrossRef]
9. Peng, Z.; Gong, X.; Lu, Z.; Xu, X.; Wei, B.; Prasad, M. A novel fabric defect detection network based on attention mechanism and multi-task fusion. In Proceedings of the 2021 7th IEEE International Conference on Network Intelligence and Digital Content (IC-NIDC), Beijing, China, 17–19 November 2021; pp. 484–488.
10. Chakraborty, S.; Moore, M.; Parrillo-Chapman, L. Automatic defect detection of print fabric using convolutional neural network. *arXiv* **2021**, arXiv:2101.00703.
11. Jing, J.; Ren, H. Defect detection of printed fabric based on RGBAAM and image pyramid. *Autex Res. J.* **2021**, *21*, 135–141. [CrossRef]
12. Zhang, J.; Jing, J.; Lu, P.; Song, S. Improved MobileNetV2-SSDLite for automatic fabric defect detection system based on cloud-edge computing. *Measurement* **2022**, *201*, 111665. [CrossRef]
13. Jia, Z.; Shi, Z.; Quan, Z.; Mei, S. Fabric defect detection based on transfer learning and improved Faster R-CNN. *J. Eng. Fibers Fabr.* **2022**, *17*, 15589250221086647. [CrossRef]
14. Sabeenian, R.S.; Paul, E.; Prakash, C. Fabric defect detection and classification using modified VGG network. *J. Text. Inst.* **2023**, *114*, 1032–1040. [CrossRef]

15. Liu, A.; Yang, E.; Wu, J.; Teng, Y.; Yu, L. Double sparse low-rank decomposition for irregular printed fabric defect detection. *Neurocomputing* **2022**, *482*, 287–297. [CrossRef]
16. Zheng, Y.; Cui, L. Defect detection on new samples with siamese defect-aware attention network. *Appl. Intell.* **2023**, *53*, 4563–4578. [CrossRef]
17. Li, L.; Li, Q.; Liu, Z.; Xue, L. Effective Fabric Defect Detection Model for High-Resolution Images. *Appl. Sci.* **2023**, *13*, 10500. [CrossRef]
18. Thesis. Thesis Dataset Dataset. Roboflow Universe. 2023. Available online: https://universe.roboflow.com/thesis-wy7ne/thesis-dataset-wfmza (accessed on 26 January 2024).
19. Istanbul Technical University. FabricDefectDet2 Dataset. Roboflow Universe. 2023. Available online: https://universe.roboflow.com/istanbul-technical-university-hygeg/fabricdefectdet2 (accessed on 26 January 2024).
20. os. defect_1 Dataset. Roboflow Universe. 2022. Available online: https://universe.roboflow.com/os-xda7q/defect_1-8gw3m (accessed on 26 January 2024).
21. Workgroup on Texture Analysis of DFG's. TILDA Textile Texture Database. 1996. Available online: http://lmb.informatik.uni-freiburg.de/resources/datasets/tilda.en.html (accessed on 16 March 2019).
22. Tianchi: Smart Diagnosis of Cloth Flaw Dataset. 2020. Available online: https://tianchi.aliyun.com/dataset/dataDetail?dataId=79336 (accessed on 15 April 2021).
23. Tianchi: Smart Diagnosis of Tile Flaw Dataset. 2020. Available online: https://tianchi.aliyun.com/dataset/dataDetail?dataId=110088 (accessed on 15 April 2021).
24. AITEX Fabric Image Database. Available online: https://www.aitex.es/afid/ (accessed on 15 April 2021).
25. Howard, A.G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; Adam, H. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *arXiv* **2017**. arXiv:1704.04861.
26. Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; Chen, L.-C. MobileNetV2: Inverted Residuals and Linear Bottlenecks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018. Available online: https://openaccess.thecvf.com/content_cvpr_2018/html/Sandler_MobileNetV2_Inverted_Residuals_CVPR_2018_paper.html (accessed on 18 February 2024).
27. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You only look once: Unified, real-time object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 26 June–1 July 2016; pp. 779–788. Available online: https://www.cv-foundation.org/openaccess/content_cvpr_2016/html/Redmon_You_Only_Look_CVPR_2016_paper.html (accessed on 18 February 2024).
28. Terven, J.; Córdova-Esparza, D.-M.; Romero-González, J.-A. A Comprehensive Review of YOLO Architectures in Computer: From YOLOv1 to YOLOv8 and YOLO-NAS. *Mach. Learn. Knowl. Extr.* **2023**, *5*, 1680–1716. [CrossRef]
29. YOLOv8 by MMYOLO. Available online: https://github.com/open-mmlab/mmyolo/tree/main/configs/yolov8 (accessed on 26 January 2024) .
30. TensorFlow. SSD MobileNet V2 FPNLite 320x320 COCO17. Available online: http://download.tensorflow.org/models/object_detection/tf2/20200711/ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8.tar.gz (accessed on 18 February 2024).