1.5em 0pt

TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING
THAPATHALI CAMPUS

PROJECT NO.: THA079MSISE06

PRINTABLE AREA MAPPING IN THE UPPER GARMENTS CONSIDERING THE OBSTACLES

BY

POSHAN KARKI

A PROJECT

SUBMITTED TO THE DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE DEGREE OF MASTER OF SCIENCE IN INFORMATICS AND INTELLIGENT SYSTEMS ENGINEERING

DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING

KATHMANDU, NEPAL

AUGUST, 2024

ACKNOWLEDGMENT

This project work would not have been possible without the guidance and the help of several individuals who in one way or another contributed and extended their valuable assistance in the preparation and completion of this study.

First of all, I would like to express my sincere gratitude to my supervisor, **Er. Umesh Kanta Ghimire**, of **IOE, Thapathali Campus** for providing invaluable guidance, insightful comments, meticulous suggestions, and encouragement throughout the duration of this project work. My sincere thanks also goes to the M.Sc. coordinator, **Er. Dinesh Baniya Kshatri**, for coordinating the project works, providing astute criticism, and having inexhaustible patience.

I am also grateful to my classmates and friends for offering me advice and moral support. To my family, thank you for encouraging me in all of my pursuits and inspiring me to follow my dreams. I am especially grateful to my parents, who supported me emotionally, believed in me and wanted the best for me.

Poshan Karki
THA079MSISE06
August, 2024

ABSTRACT

This research presents the development of an automated system to accurately classify garment types and map printable areas on men's upper-body garments using advanced deep learning techniques. The study initially focuses on classifying different garment types, followed by the implementation of Faster R-CNN with Detectron2 to detect obstacles such as zippers, buttons, pockets, and seams, and to precisely identify unobstructed printable regions. A specialized "Garments Obstacles" dataset was curated, with images meticulously annotated in Pascal VOC format and converted to COCO JSON format for training and evaluation purposes. The system's performance was rigorously assessed using metrics such as Intersection over Union (IoU), pixel-wise accuracy, precision, and recall. Results demonstrated the model's robustness in accurately classifying garment types, effectively detecting obstacles, and reliably mapping printable areas. By leveraging region proposal networks (RPN) and employing hyperparameter optimization, the study achieved a balanced approach that enhances detection accuracy while maintaining computational efficiency. The findings contribute valuable insights to the field of automated garment printing detection and have broader implications for computer vision applications.

Keywords: COCO-format Dataset, Detectron2, Garment Printing Detection, Obstacle Detection, Region Proposal Networks,

TABLE OF CONTENTS

ACKNOWLEDGMENT ..... **ii**

ABSTRACT ..... **iii**

TABLE OF CONTENTS ..... **iv**

LIST OF FIGURES ..... **viii**

LIST OF TABLES ..... **x**

LIST OF ABBREVIATIONS ..... **xi**

1 INTRODUCTION **1**

1.1 Background 1

1.2 Motivation 2

1.3 Problem Definition 3

1.4 Project Objectives 3

1.5 Scope of Project 3

1.5.1 Capabilities 3

1.5.2 Limitations 4

1.6 Potential Applications 4

1.7 Originality of Project 5

1.8 Organisation of Project Report 6

2 LITERATURE REVIEW **7**

2.1 Garment Classification and Detection 8

2.2 Obstacle Detection in Garments 9

2.3 Integration of Garment Classification and Obstacle Detection 10

2.4 Advanced Techniques in Garment Mapping 11

2.5 Research Gap 12

3 METHODOLOGY **14**

3.1 Theoretical Formulations 14

3.1.1 Faster R-CNN with Detectron2 14

3.1.2 Pre-processing Steps 22

3.1.3 Post-processing Steps 23

3.2 Advantages of Using Faster R-CNN with Detectron2 25

3.3 Assumptions in the Context of Garment Customization 26
3.4 Mathematical Modelling 28
3.4.1 Pre-processing Steps: 28
3.4.2 Faster R-CNN Model: 31
3.4.3 Post-processing Steps 32
3.5 System Block Diagram 34
3.5.1 Training Phase: 35
3.5.2 Inference Phase 37
3.6 Instrumentation Requirements 38
3.6.1 Hardware Tools: 38
3.6.2 Software Tools: 39
3.7 Dataset Explanation 40
3.7.1 Relevancy of the Dataset: 40
3.7.2 Exploration of the Dataset Contents 40
3.7.3 Dataset Preparation and Mechanism 42
3.8 Description of Algorithms 50
3.8.1 Pre-Processing Algorithms 50
3.8.2 Post-Processing Algorithms 52
3.9 Post-Processing Algorithms 55
3.9.1 Non-Maximum Suppression (NMS) 55
3.9.2 Bounding Box Refinement 55
3.9.3 Score Thresholding 56
3.9.4 Object Segmentation 56
3.9.5 Obstacle Masking 57
3.9.6 Printable Area Determination 58
3.9.7 Validation and Adjustment 58
3.9.8 Annotation Parsing 58
3.9.9 Label Encoding 59
3.10 Flowcharts and Pseudo-Code 59
3.10.1 Flowchart for Post-Processing Workflow 60
3.10.2 Pseudo-Code for Annotation Parsing and Label Encoding 60
3.11 Elaboration of Working Principle 61

3.11.1 Preprocessing of Raw Input Data 61
3.11.2 Model Manipulation Through the Stages of Faster R-CNN with FPN Backbone 64
3.11.3 Post-Processing of Model Output 66
3.11.4 Sample Calculations 68
3.12 Verification and Validation 70
3.12.1 Chosen Metrics and Their Relevance 70
3.12.2 Relevance of Chosen Metrics 70
3.12.3 Definitions and Formulas 72
4 RESULTS 74
4.1 Initial Setup and Early Findings 75
4.1.1 Experiment One with 1000 Iterations: 75
4.1.2 Experiment Two with 1000 Iterations: 80
4.1.3 Initial Findings and Performance Evaluation 84
4.1.4 Identified Limitations and Technical Challenges 85
4.1.5 Strategic Directions for Subsequent Phases 86
4.1.6 Conclusion and Path Forward 87
4.2 Results: Second Phase of Experiments 87
4.2.1 Introduction and Objectives 87
4.2.2 Experimental Setup 88
4.2.3 Experiment with k=5 and 700 Iterations: 89
4.2.4 Experiment with k=10 and 3000 Iterations: 92
4.2.5 Average Best-Case Scenario of all Folds 93
4.2.6 Worst-Case Scenario of all Folds 93
4.2.7 Obstacle-Wise Analysis 101
4.2.8 CategoryWise Output Scenerios 103
5 DISCUSSION AND ANALYSIS 111
5.1 Comparison of Benchmark and Experimental Outputs 112
5.1.1 Benchmark Overview 112
5.1.2 Experimental Results Overview 112
5.1.3 Comparison 113
5.1.4 Observations and Insights 118

5.2 Error Analysis 119
5.2.1 Sources of Error 119
5.2.2 Mitigation Strategies 121
5.3 Comparative Analysis with State-of-the-Art 122
5.3.1 Performance Metrics 122
5.3.2 Computational Efficiency 124
5.3.3 Practical Applicability 125
5.4 Performance Comparison with Existing Works 126
5.5 Improving Overall Results 129
5.6 Recommendations for Future Researchers 130
6 CONCLUSION 131
APPENDIX A
A.1 Project Schedule 132
A.2 Literature Review of Base Paper- I 133
A.3 Literature Review of Base Paper- II 134
A.4 Literature Review of Base Paper- III 135
A.5 Literature Review of Base Paper- IV 136
A.6 Literature Review of Base Paper- V 137
A.7 Mathematical Derivations 138
REFERENCES 144

LIST OF FIGURES

Figure 3.1 Architecture of Faster R-CNN [1] 14
Figure 3.2 Detailed architecture of Base-RCNN-FPN.[2] 16
Figure 3.3 Architecture of RPN Network [1] 18
Figure 3.4 Architecture of Meta's Base R-CNN FPN[2] 19
Figure 3.5 System block diagram for obstacle detection and print area mapping on garments using Faster R-CNN and Mask R-CNN. 34
Figure 3.6 Dataset showing different garment types: Hoodie, T-shirt, Jacket, Sweatshirt, Polo T-shirt, and Polo Shirt. 42
Figure 3.7 Flowchart depicting the dataset preparation and model training process 43
Figure 3.8 Garment Annotated Dataset distribution 44
Figure 3.9 Manual annotation of t-shirt and hoodie images 45
Figure 3.10 Flowchart of the Post-Processing Workflow 60
Figure 3.11 Flow Chart for obstacle detection and print area mapping on garments using Faster R-CNN and Mask R-CNN. 62
Figure 3.12 Detailed architecture of Base-RCNN-FPN [2] 64
Figure 3.13 Post-Processing Workflow 67
Figure 4.1 Training Losses Over 1000 Iterations 77
Figure 4.2 Learning Rate Schedule 78
Figure 4.3 Average Precision (AP) metrics for various thresholds and object sizes. 79
Figure 4.4 Training Losses Over 1000 Iterations 82
Figure 4.5 Learning Rate Schedule 82
Figure 4.6 Average Precision (AP) metrics for various thresholds and object sizes. 83
Figure 4.7 Training Losses Over k fold 5 and 700 Iterations 90
Figure 4.8 Learning Rate Schedule 91
Figure 4.9 Losses over the training rate in 700 Iterations 91
Figure 4.10 Average Precision (AP) metrics for various thresholds and object sizes. 92
Figure 4.11 Training Losses on fold 10 and 3000 Iterations 95
Figure 4.12 Average Training Losses Over k fold 10 and 3000 Iterations 95
Figure 4.13 Learning Rate Schedule 96
Figure 4.14 Learning Rate Schedule 97

Figure 4.15 Average Losses vs Learning Rate Over the Iterations 97
Figure 4.16 Average Precision (AP) metrics at fold 10 98
Figure 4.17 Overall Average Precision (AP) metrics for various thresholds and object sizes. 98
Figure 4.18 Categorywise Average Precision (AP) for all folds 100
Figure 4.19 Best Case Scenario 1 for Jacket 103
Figure 4.20 Best Case Scenario 2 for Jacket 103
Figure 4.21 Worst Case Scenario 1 for Jacket 104
Figure 4.22 Worst Case Scenario 2 for Jacket 104
Figure 4.23 Best Case Scenario 1 for Polo T-shirt 104
Figure 4.24 Best Case Scenario 2 for Polo T-shirt 105
Figure 4.25 Worst Case Scenario 1 for Polo T-shirt 105
Figure 4.26 Worst Case Scenario 2 for Polo T-shirt 105
Figure 4.27 Best Case Scenario 1 for Sweat 106
Figure 4.28 Best Case Scenario 2 for Sweat 106
Figure 4.29 Worst Case Scenario 1 for Sweat 106
Figure 4.30 Worst Case Scenario 2 for Sweat 106
Figure 4.31 Best Case Scenario 1 for Hoodie 107
Figure 4.32 Best Case Scenario 2 for Hoodie 107
Figure 4.33 Worst Case Scenario 1 for Hoodie 107
Figure 4.34 Worst Case Scenario 2 for Hoodie 108
Figure 4.35 Best Case Scenario 1 for Shirt 108
Figure 4.36 Best Case Scenario 2 for Shirt 108
Figure 4.37 Best Case Scenario 2 for Shirt 109
Figure 4.38 Worst Case Scenario 1 for Shirt 109
Figure 4.39 Worst Case Scenario 1 for Shirt 109
Figure 4.40 Best Case Scenario 1 for T-Shirt 110
Figure 4.41 Best Case Scenario 2 for T-Shirt 110
Figure 4.42 Worst Case Scenario 1 for T-Shirt 110
Figure 4.43 Worst Case Scenario 2 for T-Shirt 110
Figure A.1 Gantt Chart of Project Timeline. 132

LIST OF TABLES

Table 4.1 Summary of Major Experiments 74
Table 4.2 Experiment One Hyperparameter Configuration 75
Table 4.3 Performance Metrics 76
Table 4.4 Experiment Two Hyperparameter Configuration 80
Table 4.5 Performance Metrics 81
Table 4.6 Experiment One with K-cross Configuration 89
Table 4.7 Performance Matrix 90
Table 4.8 Experiment Two with K-cross Configuration 92
Table 4.9 Average Precision (AP) by Category for fold 10 94
Table 4.10 Average Precision (AP) by Category for all folds 94
Table 4.11 Performance Metrics Summary 99
Table 4.12 Average AP for each category across all folds 100
Table 4.13 Average AP for each obstacle across all folds 101
Table 4.14 Average AP for each garment across all folds 102
Table 4.15 Average AP for each printable area across all folds 102
Table 5.1 Comparison of Benchmark and Experimental Results 113
Table 5.2 Category-wise Comparison of Average Precision 115
Table 5.3 Comparison of Precision and Recall 123
Table 5.4 Comparison of mAP Scores 123
Table 5.5 Comparison of IoU Scores 124
Table 5.6 Comparison of Computational Efficiency 125

LIST OF ABBREVIATIONS

ABBR	ABBREVIATIONS
AP	Average Precision
CNN	Convolutional Neural Network
COCO	Common Objects in Context
Fast R-CNN	Fast Region-based Convolutional Neural Network
Faster R-CNN	Faster Region-based Convolutional Neural Network
FN	False Negative
FP	False Positive
GPU	Graphics Processing Unit
ICCV	IEEE International Conference on Computer Vision
IoU	Intersection over Union
JSON	JavaScript Object Notation
LR	Learning Rate
mAP	mean Average Precision
NMS	Non-Maximum Suppression
R-CNN	Region-based Convolutional Neural Network
RPN	Region Proposal Network
SSD	Single Shot MultiBox Detector
SVM	Support Vector Machine
TN	True Negative
TP	True Positive
U-Net	U-Net Convolutional Network
VOC	Visual Object Classes
YOLO	You Only Look Once

1 INTRODUCTION

1.1 Background

The traditional garment industry has long grappled with the arduous task of mapping printable areas on upper body garments, a process fraught with inefficiencies and potential errors. Conventional methods, heavily reliant on manual labor, have been time-consuming and labor-intensive. Moreover, these antiquated approaches often failed to account for the intricate nuances of garment construction, such as seams, pockets, and zippers, which acted as obstacles and led to inaccurate mapping of printable areas.

This shortcoming in traditional mapping techniques had far-reaching consequences for the production process, resulting in inefficiencies, waste, and suboptimal utilization of resources. The inability to accurately identify printable areas often led to misprints, misalignments, and other defects, ultimately compromising the quality and consistency of the final product.

In response to these challenges, the fashion industry has been undergoing a transformative shift towards automation and digitalization, driven by the increasing demand for innovative solutions in garment design and production. This paradigm shift has been catalyzed by the rapid advancement of technologies such as computer-aided design (CAD), computer vision, and machine learning algorithms.

CAD systems, coupled with advanced pattern-making software, have revolutionized the mapping process by providing virtual 3D representations of garments. These digital models could be manipulated, analyzed, and optimized to precisely identify printable areas while accounting for the intricate geometric constraints imposed by seams, pockets, and zippers. By leveraging computational power and sophisticated algorithms, these CAD systems streamlined the mapping process, reducing manual labor and minimizing the potential for human error.

Furthermore, the integration of computer vision and machine learning techniques has automated the mapping process even further. These technologies were trained on vast datasets of garment designs and patterns, enabling them to recognize and accurately map printable areas with unprecedented accuracy and speed. Machine learning algorithms continuously refined their mapping capabilities, ensuring consistent, high-quality results.

The confluence of these technological advancements has revolutionized the garment industry, enabling efficient and precise mapping of printable areas, minimizing waste, and optimizing resource utilization. By embracing automation and digitalization, the fashion industry has unlocked new realms of productivity, quality, and sustainability, paving the way for a more efficient and eco-friendly future in garment design and production.

1.2 Motivation

As a student of computer vision and machine learning and a professional in the software field, I have been driven by the ongoing digitalization and automation in various industries, including e-commerce and fashion. I recognized that an automated printable area mapping system could address the existing limitations of traditional mapping techniques for garments, leveraging the power of cutting-edge technologies that I have been actively studying and applying.

Conventional manual mapping methods employed in the fashion industry were labor-intensive, time-consuming, and susceptible to human errors, often leading to inaccuracies and inefficiencies in the production process. As an ardent learner of computer vision and machine learning, I recognized the potential of these technologies to offer a more consistent, objective, and automated approach to assessing garment patterns and identifying printable areas while intelligently accounting for obstacles such as seams, pockets, and zippers.

By combining my academic pursuits in computer vision and machine learning with my professional experience in the software domain, I was motivated to develop an automated system that could serve as a screening tool for apparel design and production planning. Such a system could accelerate the development of printed garments, reduce subjective biases inherent in manual techniques, and uncover new frontiers in garment design and printing, ultimately minimizing the potential for defects and waste.

With the successful development of an automated printable area mapping system and its demonstration of parallel validity with current industry standards, fashion designers and manufacturers can now leverage this technology to streamline their mapping processes with objective assessments. This not only enhances efficiency and accuracy but also

facilitates remote mapping services, overcoming geographical barriers and enabling access to efficient mapping solutions in resource-constrained regions.

As a software professional with a keen interest in computer vision and machine learning, I have been driven by the opportunity to apply my knowledge and skills to address real-world challenges in the e-commerce and fashion industries. By developing this automated printable area mapping system, I hope to contribute to the ongoing digitalization efforts, streamlining processes, reducing waste, and unlocking new possibilities in garment design and production.

1.3 Problem Definition

Traditional methods of mapping printable areas on upper body garments lacked the sophistication to handle obstacles effectively, leading to inaccuracies and inefficiencies in the production process. There was a pressing need for an automated system that could classify garment types, intelligently detect obstacles such as seams, pockets, and zippers, and accurately map printable areas according to the standard print area specifications of the garment. This research successfully developed such a system using Faster R-CNN, enhancing the precision and reliability of obstacle detection and printable area identification, thereby streamlining the design and production workflow in the textile industry.

1.4 Project Objectives

- To develop a robust system for classifying garment types, detecting obstacles such as zippers and pockets, and detecting the obstacle-free printable areas on male upper-body garments using Faster R-CNN.
- To achieve high accuracy in identifying obstacles and printable regions on garments, leveraging a curated comprehensive dataset and advanced image augmentation techniques.

1.5 Scope of Project

1.5.1 Capabilities

This project focuses on developing an advanced system for classifying male upper-body garments and accurately mapping printable areas by utilizing the Faster R-CNN

architecture with Detectron2. The system is designed to handle various garment types, including shirts, t-shirts, hoodies, jackets, polo t-shirts, sweaters, and sweatshirts, and is capable of detecting obstacles such as seams, pockets, zippers, and buttons. By identifying these obstacles and adhering to standard specifications for printable areas, the system aims to optimize the garment design and production process, minimizing manual errors and enhancing overall efficiency. The project leverages a custom-curated dataset named "Garments Obstacles," which is supplemented with advanced data augmentation techniques to improve the model's accuracy and reliability, ensuring robust performance in detecting and classifying garment types and obstacles.

1.5.2 Limitations

Despite its advanced capabilities, the project encounters several limitations that affect the system's performance and generalizability. A significant challenge lies in the relatively small, self-prepared dataset, which may introduce biases and limit the model's ability to accurately detect and classify various obstacles, particularly small or subtle features like buttons and logos. Additionally, while Faster R-CNN is effective for object detection, it may not be the most suitable choice for attribute mapping tasks, such as identifying left chest, right chest, and front center printable areas, as these are not distinct objects but rather regions within garments that often appear as blank spaces. This reliance on rectangular bounding boxes for annotations can lead to inaccuracies and discrepancies, especially when dealing with complex garment designs. The project also faces scalability challenges due to the significant computational demands of training the Faster R-CNN model, which may be prohibitive for smaller production environments with limited resources. Moreover, the focus on male upper-body garments excludes sleeves, back parts, and other garment types, thereby limiting the model's applicability to a broader range of garment styles and scenarios. To overcome these limitations, future work should explore expanding the dataset, adopting more advanced model architectures like Mask R-CNN, and utilizing more precise annotation methods, such as mask or polygon annotations, to improve the system's accuracy, robustness, and generalizability.

1.6 Potential Applications

The system developed for Automated Printable Area Mapping on Upper Body Garments Considering Obstacles has the potential to revolutionize the fashion industry by stream-

lining the garment design and production process. By accurately identifying printable regions on garments while accounting for obstacles, the system enables efficient and cost-effective customization of garments with printed designs, logos, or patterns. This opens up new avenues for personalization and mass customization in the fashion industry, catering to diverse consumer preferences and enabling on-demand production.

Beyond the fashion industry, the system may also find applications in other domains that require accurate mapping of complex surfaces, such as:

1. **Medical Textiles:** The system can be adapted to map printable areas on medical garments, bandages, or wearable devices, enabling the integration of therapeutic or diagnostic elements through printing.

2. **Automotive Upholstery:** The accurate mapping of printable areas on automotive upholstery facilitates customized designs and branding, enhancing the aesthetic appeal and personalization of vehicle interiors.

3. **Product Labeling and Branding:** The system's ability to map printable areas on various surfaces streamlines the application of labels, logos, or branding elements on a wide range of products, from consumer goods to industrial equipment.

4. **Textile Art and Crafts:** Artists and craftspeople working with textiles benefit from the system's capabilities, enabling precise printing of intricate designs or patterns on garments or fabric surfaces.

1.7 Originality of Project

This project presents a novel approach to garment mapping that addresses the limitations of existing methods by intelligently considering obstacles such as seams, pockets, and zippers. The developed system integrates cutting-edge computer vision and machine learning algorithms to enhance accuracy and efficiency in printable area mapping. By leveraging these advanced technologies, this project makes a significant contribution to the field of garment design and production.

1.8 Organisation of Project Report

This project report consists of five chapters, each focusing on automated printable area mapping on upper body garments, dealing with obstacles like seams, pockets, and buttons.

Chapter 1: Introduction introduces the project's goals and significance in simplifying garment design and customization.

Chapter 2: Historical Context explores the history of garment mapping, providing context for our work.

Chapter 3: Methodology explains our research methods, including diagrams and algorithms used in our system.

Chapter 4: Results and Discussion presents the results of the project and discusses their implications for the fashion industry.

Chapter 5: Appendices contains supplementary materials such as code snippets and datasets for reference.

2 LITERATURE REVIEW

The integration of automated garment mapping technologies within the fashion industry is poised to significantly enhance design precision and streamline production workflows. However, despite the promise of these technologies, accurately identifying hassle-free printable areas on garments, especially in the presence of obstacles like seams, pockets, and zippers, presents a complex challenge.

Recent advancements in the field have increasingly relied on sophisticated convolutional neural networks (CNNs) and region-based architectures, which are well-suited to the intricate task of garment analysis. Among these, the Faster R-CNN architecture has emerged as a leading approach due to its ability to efficiently detect and localize garment features and obstacles with high precision. The architecture's dual-stage framework—comprising a Region Proposal Network (RPN) and a Fast R-CNN detector—enables it to perform robust object detection, making it particularly effective for complex tasks within the garment industry.

Building on the foundational strengths of Faster R-CNN, several researchers have proposed enhancements tailored to the specific demands of garment analysis. For instance, integrating a Feature Pyramid Network (FPN) into Faster R-CNN has been shown to significantly improve multi-scale feature representation, a critical factor for accurately detecting smaller garment features such as buttons, zippers, and intricate seam patterns. This enhancement not only improves detection accuracy but also addresses the challenge of varying garment sizes and styles, which often complicate the detection process.

Additionally, the application of Mask R-CNN—a further extension of Faster R-CNN—has proven effective in addressing the need for pixel-wise segmentation of printable areas on garments. By adding a segmentation branch, Mask R-CNN is capable of generating detailed masks that delineate both obstacles and potential printing zones, providing a granular view of garment surfaces that is essential for high-precision printing applications.

Despite these significant advancements, several challenges persist. The high variability in garment styles, materials, and printing techniques continues to pose a considerable

obstacle to the generalizability of current systems. Moreover, the need for real-time processing in industrial production environments remains largely unmet by existing methodologies, many of which are computationally intensive and struggle to deliver the necessary speed without compromising accuracy.

To address the scarcity of large-scale, annotated datasets specific to garment mapping and obstacle detection, researchers have begun to explore transfer learning and self-supervised learning techniques. These approaches leverage pre-existing models and unlabeled data to build more robust systems capable of adapting to the diverse and dynamic nature of garment production.

2.1 Garment Classification and Detection

Garment classification and detection serve as the foundation of automated garment mapping, which is essential for customizing and personalizing garments in the fashion industry. Traditional methods for garment classification typically relied on manual annotation and classical computer vision techniques such as edge detection and template matching. While these methods were useful in specific controlled environments, they were often time-consuming, labor-intensive, and prone to inaccuracies when applied to diverse and complex datasets.

Li et al. [3] proposed an innovative approach that integrated classical computer vision techniques with modern machine learning algorithms to improve the accuracy of garment classification. Their approach combined feature extraction methods like Scale-Invariant Feature Transform (SIFT) and Histogram of Oriented Gradients (HOG) with a support vector machine (SVM) classifier. While their method improved the accuracy of identifying garment types and printable areas, it was limited by the need for further validation on diverse datasets, particularly in real-world scenarios where the variability in garment appearance can be significant.

The advent of deep learning marked a significant advancement in garment classification. Velswamy and Delight [4] leveraged Mask R-CNN, a deep learning-based object detection and instance segmentation framework, to enhance garment mapping accuracy. Mask R-CNN extends Faster R-CNN by adding a branch for predicting segmentation masks, allowing for precise detection and classification of individual garment components.

Their approach achieved a mean Average Precision (mAP) of 78.6% across various garment types, demonstrating significant improvements over traditional methods. However, the authors noted that the computational complexity of Mask R-CNN, combined with the need for substantial training data, presented challenges in resource-constrained environments where high computational power is not always available.

In another notable work, Zhang et al. [5] applied classical computer vision techniques to the task of garment classification and obstacle detection. Their approach relied on edge detection and contour analysis to identify garment boundaries and obstacles. Although their method was computationally efficient and required less data, it struggled with the robustness needed to handle complex garment designs, diverse obstacle scenarios, and variations in lighting and fabric texture.

2.2 Obstacle Detection in Garments

Obstacle detection is a critical component in garment mapping, as it directly influences the accuracy of identifying printable areas. Obstacles such as seams, zippers, buttons, and pockets can interfere with the printing process, leading to defects and inaccuracies in the final product. Accurate detection of these obstacles is therefore essential for successful garment customization.

Wang et al. [6] presented a deep learning-based method specifically designed for seam detection in garment images. They employed a convolutional neural network (CNN) architecture with multiple convolutional layers followed by fully connected layers to detect seams with high accuracy. Their model achieved a precision of 85.4% and a recall of 82.7% on a custom dataset of garment images. The authors highlighted the robustness of their method in detecting seams under various lighting conditions and fabric textures, but they also noted that the computational demands of their approach posed limitations for real-time applications, which are crucial in a high-throughput production environment.

Building on this work, Chen et al. [7] focused on pocket detection, another common obstacle in garment printing. They utilized a deep CNN architecture similar to that of Wang et al., but with additional layers designed to capture the unique features of pockets, such as their shape and texture. Their method achieved a precision of 88.9% and a recall

of 86.3% in detecting pockets across various garment types. The authors emphasized the importance of accurate pocket detection for avoiding printing errors, particularly in garments where pockets are prominent features. However, they also pointed out that their method required further validation on more diverse datasets to assess its generalizability and robustness in different environmental conditions.

Zhang et al. [5] explored an alternative approach by integrating obstacle detection with garment mapping using classical computer vision techniques. Their method involved using edge detection and contour analysis to identify obstacles such as zippers and buttons. While their approach was computationally efficient and required less data than deep learning methods, it lacked the robustness needed to handle complex garment designs and diverse obstacle scenarios. The authors acknowledged that their reliance on non-deep learning methods limited the accuracy and generalizability of their approach, particularly in more challenging garment mapping tasks.

2.3 Integration of Garment Classification and Obstacle Detection

The integration of garment classification with obstacle detection is crucial for creating a comprehensive automated garment mapping system. This integration ensures that the system not only classifies the garment type but also accurately identifies and maps out printable areas by detecting and accounting for obstacles.

The use of region-based convolutional neural networks (R-CNN) has proven effective in integrating these tasks. For example, Faster R-CNN has gained attention for its ability to combine object detection with classification in a unified framework. The architecture of Faster R-CNN includes a Region Proposal Network (RPN) that shares full-image convolutional features with the detection network, allowing for faster and more accurate detection of multiple objects, including obstacles, within an image.

Wang et al. [6] extended the Faster R-CNN framework to detect garment landmarks, such as seams and hems, which are critical for fine-grained fashion analysis and obstacle detection. Their approach showed promising results, particularly in identifying key landmarks that could serve as obstacles in the printing process. The precision and recall for landmark detection were reported at 83.2% and 81.7%, respectively. However, their method's performance varied depending on the garment's complexity and the diversity

of its appearance, indicating a need for further refinement to improve its generalizability.

In the context of garment mapping, the integration of obstacle detection using Faster R-CNN can significantly improve the accuracy of printable area identification. By simultaneously classifying the garment type and detecting obstacles, the system can ensure that only unobstructed areas are identified as suitable for printing. This approach not only improves the accuracy of the mapping process but also reduces the likelihood of defects during production, which is crucial for maintaining high-quality standards in garment manufacturing.

2.4 Advanced Techniques in Garment Mapping

Beyond the basic integration of garment classification and obstacle detection, advanced techniques such as instance segmentation, multi-task learning, and transfer learning have been explored to further enhance the accuracy and efficiency of garment mapping systems.

Instance segmentation, as employed by Velswamy and Delight [4], involves identifying and delineating each object within an image, allowing for more precise mapping of obstacles and printable areas. This technique is particularly useful in garments with complex patterns and multiple overlapping obstacles, where traditional detection methods may struggle. Their method achieved a significant improvement in segmentation accuracy, with an Intersection over Union (IoU) score of 76.3%. However, the authors pointed out the increased computational requirements as a limitation, particularly for real-time applications.

Multi-task learning, where a model is trained on multiple related tasks simultaneously, has also shown promise in garment mapping. Xu et al. [8] proposed a multi-task learning framework that simultaneously tackled garment classification, obstacle detection, and landmark identification. By sharing knowledge across these tasks, their model achieved better generalization and improved overall accuracy, with an average precision of 84.7% across tasks. The main challenge identified by the authors was the need for extensive hyperparameter tuning to balance the performance across different tasks, which could be time-consuming and computationally expensive.

Transfer learning, which involves fine-tuning a pre-trained model on a new dataset, has been used to adapt existing garment mapping models to new garment types and obstacle scenarios. Zhao et al. [9] used a pre-trained Faster R-CNN model and fine-tuned it on a custom dataset of garments with various obstacles. This approach led to a 10% improvement in obstacle detection accuracy compared to training from scratch. The authors emphasized the efficiency of transfer learning in reducing the training time and the amount of labeled data required. However, they also noted that transfer learning might not always be effective when the target dataset differs significantly from the source dataset, highlighting the need for careful selection of pre-trained models.

Additionally, Ku et al. [10] introduced an automated sewing system enabled by machine vision for smart garment manufacturing. Their comprehensive system integrated machine vision with automation techniques to enhance manufacturing processes, achieving substantial improvements in production efficiency and accuracy. While their system showcased the potential of machine vision technologies in garment manufacturing, challenges such as scalability and adaptability to various garment types and production environments remain areas for further research.

2.5 Research Gap

Despite significant advancements in deep learning methods for garment classification and obstacle detection, a critical gap remains unaddressed in the current body of research—integrating these tasks into a unified system that can accurately classify male upper-body garments, detect obstacles, and identify obstacle-free print areas based on established industry standards.

Existing research has typically focused on individual aspects of garment processing, such as classification or obstacle detection, but has not fully explored the integration of these tasks. Most systems are trained on datasets that lack the diversity necessary to represent the wide range of male upper-body garments and the various obstacles that can interfere with printing. This limitation hampers the generalizability of these models to new garment types and different production conditions, where variations in fabric, design, and fit can significantly impact detection accuracy.

Moreover, there is a notable absence of research that aligns the identification of printable

areas with industry-specific guidelines, such as those outlined in the Stahls' Design Placement Guide [11]. This guide defines precise locations on garments for applying prints, which is crucial for maintaining quality and consistency in production. The lack of integrated systems that address garment classification, obstacle detection, and the identification of obstacle-free standard print areas limits the practical application of existing methods in real-world manufacturing and printing environments.

Furthermore, the computational demands of deep learning models, particularly those used in instance segmentation, present challenges for real-time applications in the fashion industry, where speed and efficiency are critical. While there has been progress in optimizing models for accuracy, balancing this with computational efficiency remains a significant challenge.

Given these gaps, your research aims to pioneer the development of a comprehensive system that not only classifies male upper-body garments and detects obstacles but also identifies obstacle-free print areas according to industry standards. This integrated approach is novel and essential for advancing the practical application of automated garment processing and printing technologies in the fashion industry.

Addressing these research gaps—by developing a unified, efficient, and scalable system—will contribute to the field by providing a robust solution that can be widely adopted in real-world scenarios, ultimately enhancing the efficiency and reliability of garment processing and printing.

3 METHODOLOGY

3.1 Theoretical Formulations

3.1.1 Faster R-CNN with Detectron2

Faster R-CNN [12] is a prominent object detection framework that integrates a Region Proposal Network (RPN) to enhance the detection process. Faster R-CNN [12] is a two-stage object detection model that consists of two main components: the Region Proposal Network (RPN) and the Fast R-CNN detector. The RPN is a fully convolutional network that generates region proposals, which are rectangular bounding boxes likely to contain objects of interest. The RPN takes an input image and outputs a set of object proposals, each with an objectness score representing the likelihood of an object being present in that region.

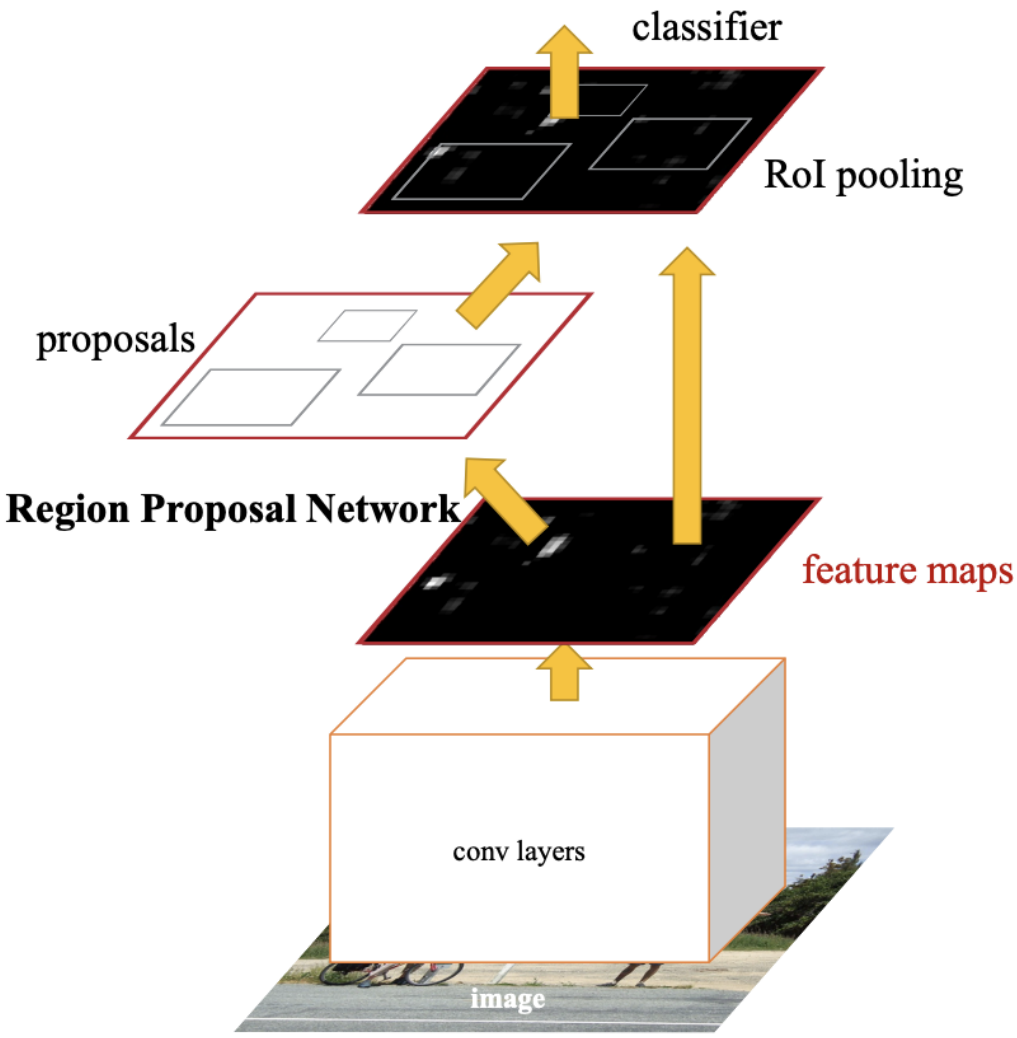


Figure 3.1: Architecture of Faster R-CNN [1]

The Fast R-CNN detector is responsible for classifying the objects within the proposed regions and refining the bounding box coordinates. It takes the region proposals from the RPN and extracts features from each proposal using a convolutional neural network (CNN). These features are then fed into two separate fully connected layers: one for object classification and another for bounding box regression.When implemented using Detectron2, a high-performance library from Facebook AI Research, Faster R-CNN benefits from optimized components and cutting-edge features.

Faster R-CNN's architecture comprises several key components:

1. **Backbone Network**

   In Detectron2, the backbone architecture serves as the foundational feature extractor, transforming raw pixel data into a hierarchical set of feature maps. These feature maps encapsulate various levels of abstraction, from low-level details like edges and textures to high-level semantic information such as object parts. The backbone is typically a deep convolutional neural network (CNN), organized into stages, each comprising multiple convolutional layers. These stages progressively reduce the spatial resolution of the input image while simultaneously increasing the depth and complexity of the feature representation.

   Some common backbone architectures are:

   - **ResNet Variants**: ResNet architectures, such as ResNet-50 and ResNet-101, use residual connections that enable the training of very deep networks by allowing gradients to flow directly through the network, bypassing some layers. This mitigates the vanishing gradient problem [13], ensuring that even the early layers of the network are trained effectively.
   - **ResNeXt**: This architecture extends ResNet by incorporating grouped convolutions, which partition the feature maps into smaller groups. Each group undergoes separate convolutions, leading to enhanced computational efficiency and increased model capacity without a proportional increase in computational cost.
   - **FBNet**: Designed for mobile and resource-constrained environments, FBNet uses neural architecture search (NAS) to optimize the network structure for

efficient inference on mobile devices. It balances accuracy and efficiency by selecting the most appropriate building blocks (e.g., convolution types, activation functions) for each layer.

The backbone's output consists of a series of feature maps, denoted as $\{P_2, P_3, P_4, P_5\}$, corresponding to different stages in the network. The resolution of these feature maps decreases progressively, with P_2 having the highest resolution and P_5 the lowest, typically halving the spatial dimensions at each stage (i.e., P_2 has a stride of 4, P_3 of 8, and so on).

2. **Feature Pyramid Network (FPN)**

   The Feature Pyramid Network (FPN) is a critical component in Detectron2 that addresses the challenge of detecting objects across a wide range of scales. It enhances the backbone's output by generating a multi-scale feature representation, allowing the model to detect both small and large objects effectively at multiple scales, improving the detection of objects of varying sizes [14].

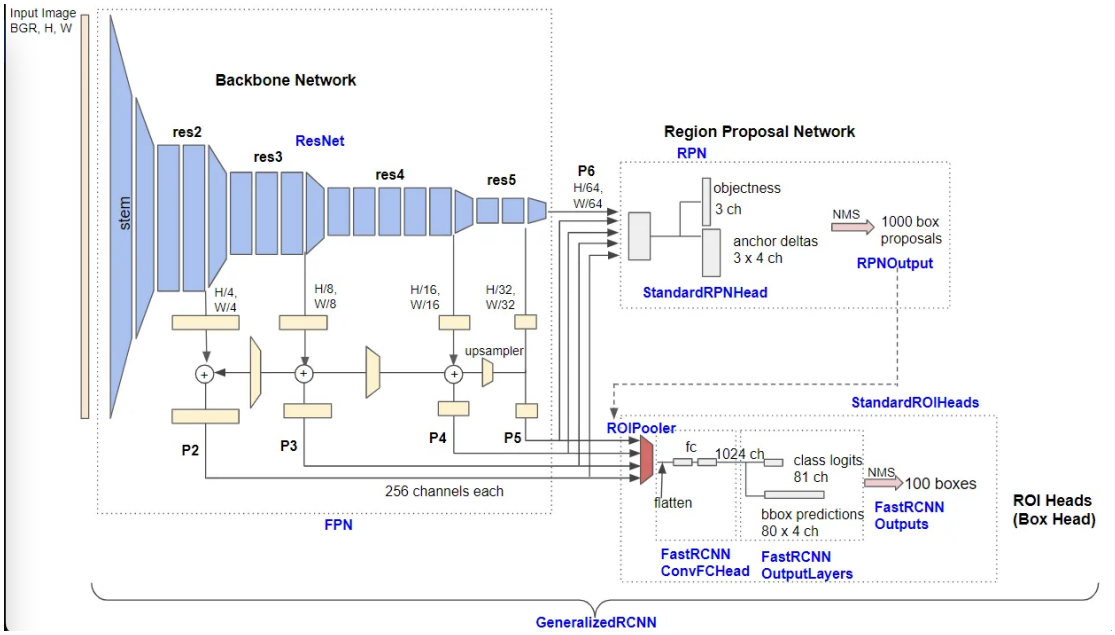


Figure 3.2: Detailed architecture of Base-RCNN-FPN.[2]

3. **FPN Architecture Components**

   - **Bottom-Up Pathway**: This pathway is essentially the forward pass through the backbone network, producing the initial set of feature maps $\{P_2, P_3, P_4, P_5\}$.

These feature maps capture visual information at varying levels of abstraction, with P_5 being the most semantically rich but spatially coarse.

- **Top-Down Pathway**: To construct higher-resolution feature maps, the top-down pathway begins at the coarsest feature map, P_5. It uses upsampling operations (typically by a factor of 2) to increase the spatial resolution, creating feature maps $\{P_5, P_4, P_3, P_2\}$. Each upsampled feature map is then refined by adding the corresponding feature map from the bottom-up pathway after it has been processed by a 1×1 convolution, ensuring that high-resolution details are incorporated.
- **Lateral Connections**: To effectively merge the top-down and bottom-up pathways, lateral connections are employed. These are 1×1 convolutions applied to the feature maps from the bottom-up pathway before they are combined with the upsampled features. This process helps preserve the spatial structure and mitigates potential aliasing effects from the upsampling.

The final output of the FPN is a set of feature maps $\{P_2, P_3, P_4, P_5\}$, each with a consistent channel dimension (typically 256). These feature maps provide a rich, multi-scale representation of the input image, enabling downstream components like the Region Proposal Network (RPN) and ROI heads to detect objects at their most appropriate scale.

The FPN's design allows Detectron2 to handle objects of various sizes with greater accuracy, improving the model's robustness across different scenarios. Additionally, Detectron2's implementation of FPN can be tailored by adjusting parameters such as the number of levels, the dimensionality of the feature maps, and the specific types of convolutions used. This flexibility enables fine-tuning for specific tasks or deployment environments.

4. **Region Proposal Network (RPN)**

The Region Proposal Network (RPN) is a critical component that theoretically addresses the problem of efficiently identifying regions of interest (RoIs) within the high-dimensional space of an image. From an information-theoretic perspective, the RPN functions as a form of attention mechanism, guiding computational resources towards areas of the image that are most likely to contain objects.

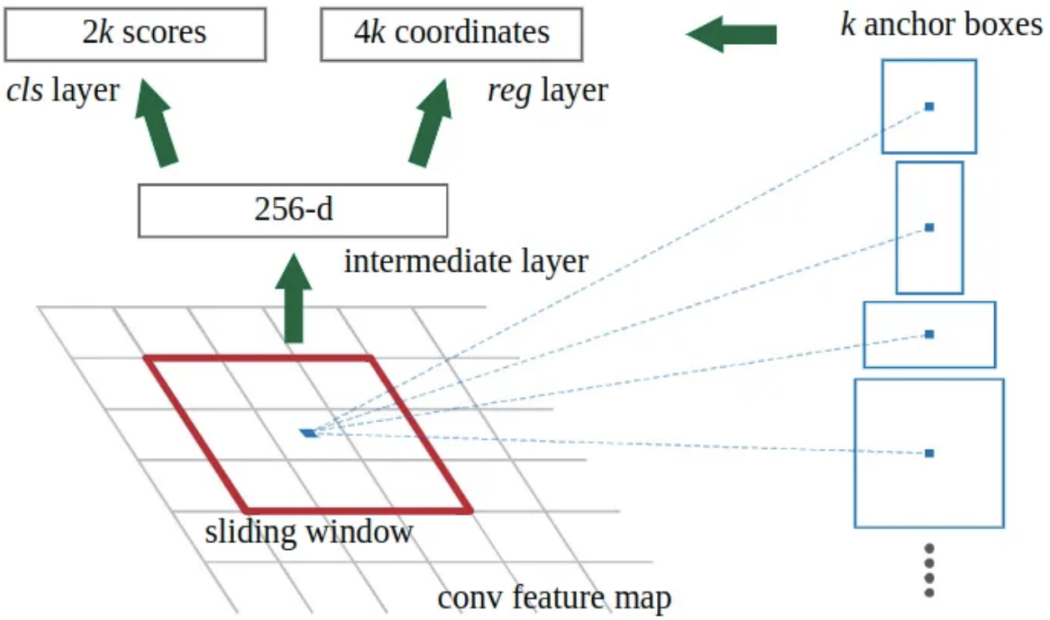


Figure 3.3: Architecture of RPN Network [1]

The RPN implements a divide-and-conquer strategy, breaking down the complex problem of object detection into two simpler sub-problems:

- **Identifying potential object locations:** The RPN uses **anchor boxes**, which are predefined bounding boxes of different scales and aspect ratios, as a prior over object shapes and sizes. These anchors regularize the proposal generation process by providing a structured hypothesis space for possible object locations.
- **Classifying and refining these locations:** The network learns to classify these anchors as either foreground (object) or background (non-object) and refines their coordinates to better fit the object boundaries.The RPN outputs an **objectness score** and bounding box coordinates for each anchor box.

The anchor-based mechanism can be viewed as imposing a structured inductive bias on the model, leveraging prior knowledge about the distribution of object shapes and scales.

5. **ROI Align** ROI Align addresses a fundamental problem in object detection: extracting fixed-size feature maps from regions of arbitrary size while preserving spatial correspondence. Unlike ROI Pooling, which introduces quantization errors

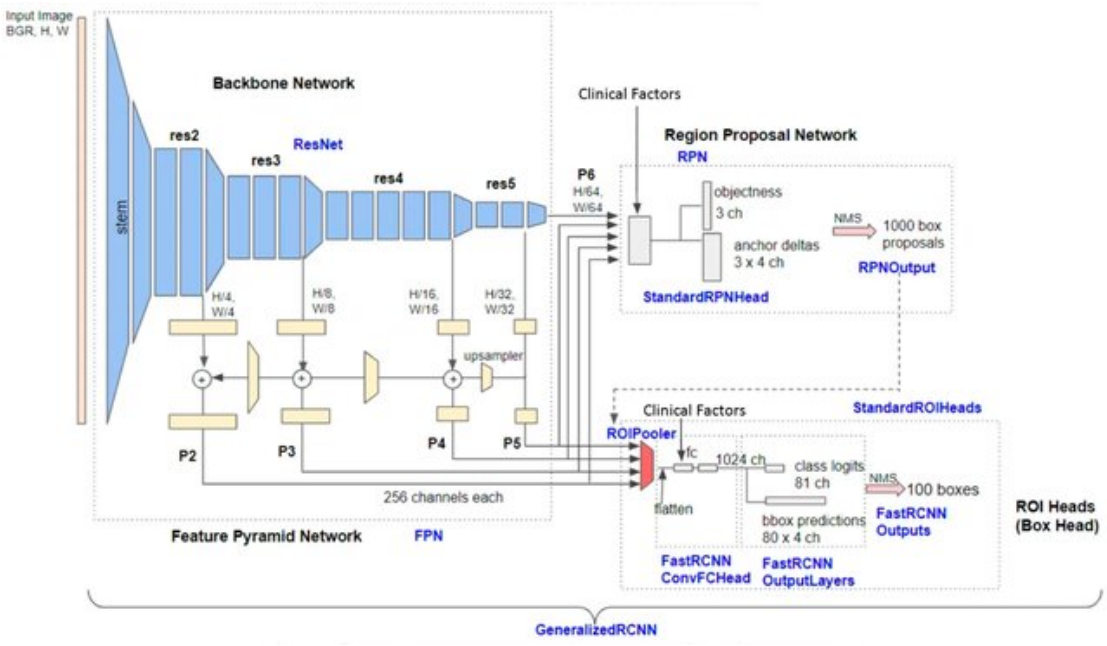


Figure 3.4: Architecture of Meta's Base R-CNN FPN[2]

by rounding coordinates, ROI Align uses bilinear interpolation to ensure that the sampling grid aligns exactly with the input feature map. Theoretically, ROI Align can be understood as a differentiable, spatially-aware sampling operation that maps irregular regions into a regular grid, allowing the network to retain fine-grained spatial information that is critical for accurate object localization and classification. ROI Align improves precision by avoiding quantization, using bilinear interpolation to better align features [1].

6. **Box Head** The box head implements the principle of hierarchical refinement, taking the coarse proposals from the RPN and applying more specialized, context-aware processing to make final predictions. This two-stage approach allows the network to progressively refine its estimates, leading to higher accuracy than single-stage detectors. Theoretically, the box head can be seen as an iterative optimization process, where each stage refines the prediction by incorporating additional contextual information. This approach aligns with the broader concept of coarse-to-fine modeling, where complex tasks are decomposed into simpler sub-tasks that are solved in a sequence of increasingly refined stages.

7. **Mask Head** The mask head extends the object detection framework to pixel-level predictions, addressing the more complex task of instance segmentation.

Theoretically, the mask head operates on the principle that instance-level features used for bounding box prediction also contain sufficient information for fine-grained segmentation. This reflects the idea of shared representations in multi-task learning, where a single network learns to perform multiple related tasks by sharing intermediate feature representations. The mask head can also be seen as an example of pixel-wise classification, where each pixel within a region is classified as belonging to the foreground object or the background, leveraging the spatial context provided by the bounding box.

8. **Keypoint Head** The keypoint head is specialized for tasks like human pose estimation, where the goal is to predict the locations of specific keypoints (e.g., joints) on the human body. Theoretically, the keypoint head can be viewed as a structured prediction task, where the network is required to predict a set of interdependent outputs (keypoints) that are spatially and semantically related. This approach leverages the idea that the features used for detecting an object can also inform the localization of its constituent parts. The keypoint head exemplifies the principle of part-based modeling, where the overall structure of an object is inferred by detecting and localizing its key components.

9. **Loss Functions** The loss functions used in object detection systems like Detectron2 are grounded in the principle of joint optimization. The network simultaneously optimizes for multiple related tasks—such as classification, localization, and segmentation—by minimizing a composite loss function. Theoretically, this approach encourages the network to learn more robust and generalizable features that are useful across different tasks. Each component of the loss function is designed to reflect the specific nature of the task:

    - **Classification Loss:** Typically implemented as a cross-entropy loss, it is used to train the network to correctly classify the object within each proposed region.
    - **Localization Loss:** Often formulated as a smooth L1 loss, it penalizes the network for inaccurate bounding box predictions, encouraging precise localization of objects.

- **Segmentation Loss:** In the case of the mask head, a pixel-wise binary cross-entropy loss is used to train the network to accurately predict the object mask.

The combination of these losses reflects a theoretical approach to balancing the different objectives during training, ensuring that the network learns to prioritize tasks based on their relative importance and difficulty.

10. **Inference Pipeline** The inference pipeline in object detection systems, including processes like non-maximum suppression (NMS) and score thresholding, addresses the challenge of converting dense, overlapping predictions into a coherent set of distinct object detections. Theoretically, NMS can be viewed as a clustering algorithm in the space of bounding boxes and confidence scores, where the goal is to eliminate duplicate detections while retaining the most confident predictions. This process can also be understood through the lens of decision theory, where the network must balance the trade-off between precision (avoiding false positives) and recall (capturing all true positives) to produce the most reliable set of detections.

11. **Cascade Stages** Cascade R-CNN and similar approaches implement the principle of curriculum learning, where the network is trained to progressively tackle increasingly difficult versions of the same task. Theoretically, this staged approach allows the network to learn more effectively by first mastering easier sub-tasks before moving on to more challenging ones. In the context of object detection, this means that the network first learns to make predictions at lower IoU thresholds, where the task is easier, and then refines these predictions at higher IoU thresholds, where the task is more challenging. This approach aligns with the concept of multi-stage optimization, where a complex problem is solved through a sequence of increasingly refined stages, leading to more accurate and robust models.

12. **Panoptic Segmentation Head** Panoptic segmentation unifies instance segmentation and semantic segmentation, addressing a fundamental dichotomy in scene understanding: the distinction between countable objects (“things”) and amorphous regions (“stuff”). Theoretically, panoptic segmentation represents a comprehensive

approach to image parsing, where the goal is to produce a single coherent interpretation of the scene that includes both discrete objects and continuous regions. This approach bridges the gap between instance and semantic segmentation, providing a more holistic understanding of the visual world by combining the strengths of both paradigms into a unified framework.

3.1.2 Pre-processing Steps

Preprocessing in object detection systems plays a critical role in preparing the input data for effective learning. The preprocessing steps are designed to enhance the input data's quality, standardize its format, and augment the dataset to improve the model's robustness and generalization.

1. **Data Augmentation:** Data augmentation is a theoretical cornerstone in the preprocessing pipeline, grounded in the concept of expanding the training set by applying transformations to existing data. These transformations—such as random cropping, flipping, rotation, color jittering, and scaling—introduce variability in the training data, effectively simulating a larger dataset. Theoretically, this process introduces invariance to certain transformations, helping the model generalize better to unseen data. Data augmentation can be understood as a regularization technique that reduces overfitting by encouraging the model to learn features that are robust to perturbations in the input data.

2. **Normalization and Standardization:** Normalization and standardization are essential preprocessing steps that ensure the input data has consistent statistical properties. Normalization involves scaling pixel values to a fixed range, typically [0, 1], while standardization adjusts the data to have zero mean and unit variance. Theoretically, these steps stabilize the training process by ensuring that the input data is on a consistent scale, preventing issues like vanishing or exploding gradients. This can be related to the concept of conditioning in optimization, where the goal is to present the optimizer with a well-conditioned problem that converges more quickly and reliably.

3. **Anchor Box Generation:** Anchor box generation is a preprocessing step that lays the groundwork for region proposal networks (RPNs) by creating a set of

predefined bounding boxes of various sizes and aspect ratios. Theoretically, anchor boxes represent a prior distribution over the space of possible object locations and scales. This prior acts as a form of inductive bias, guiding the model to focus on regions of the image that are more likely to contain objects. The anchor boxes are designed to cover the full range of object scales and aspect ratios, ensuring that the network can handle objects of varying sizes and shapes.

4. **Image Resizing and Aspect Ratio Preservation:** Image resizing is a necessary preprocessing step to ensure that all input images are of a consistent size, which is required for batch processing in convolutional neural networks (CNNs). However, resizing must be done carefully to preserve the aspect ratio of the original image, preventing distortion that could degrade the model's performance. Theoretically, this step can be viewed as a trade-off between computational efficiency and preservation of spatial information. Techniques like padding are often used to maintain the aspect ratio while resizing, ensuring that the resized images remain representative of the original scenes.

3.1.3 Post-processing Steps

Postprocessing in object detection systems is crucial for refining the raw outputs of the model into meaningful and interpretable results. These steps are designed to filter, refine, and aggregate the model's predictions, ensuring that the final output meets the desired accuracy and robustness.

1. **Non-Maximum Suppression (NMS):** Non-maximum suppression (NMS) is a key postprocessing technique used to eliminate redundant bounding box predictions. Theoretically, NMS can be viewed as a clustering algorithm in the space of bounding boxes, where the goal is to retain only the most confident predictions while suppressing overlapping boxes. This process is grounded in decision theory, where the network must balance the trade-off between precision (avoiding false positives) and recall (capturing all true positives). The NMS algorithm works by iteratively selecting the highest-scoring box and removing all boxes with significant overlap (as defined by the Intersection over Union, or IoU) with the selected box.

2. **Bounding Box Regression Refinement** Bounding box regression refinement is a postprocessing step where the initial bounding box predictions are adjusted to better fit the objects in the image. This step is theoretically rooted in the concept of iterative optimization, where the network makes coarse predictions in the early stages and refines them in later stages. The refinement process involves adjusting the position, size, and aspect ratio of the bounding boxes based on the output of the regression head, leading to more accurate localization of objects.

3. **Score Thresholding** Score thresholding is a postprocessing step used to filter out low-confidence predictions. Theoretically, score thresholding can be seen as a decision-making process where the model must determine which predictions are reliable enough to be included in the final output. By setting a confidence score threshold, the system discards predictions that fall below a certain confidence level, reducing the number of false positives. This process is akin to hypothesis testing, where only predictions that pass a certain confidence criterion are accepted as true detections.

4. **Soft-NMS and IoU-based Adjustments** Soft-NMS is an extension of the traditional NMS algorithm, where instead of completely discarding overlapping boxes, the scores of overlapping boxes are gradually reduced based on their IoU with the highest-scoring box. Theoretically, Soft-NMS addresses the issue of abrupt score changes in traditional NMS, leading to more graceful and accurate suppression of redundant boxes. IoU-based adjustments can also be used to refine the final set of bounding boxes, ensuring that the selected boxes are both highly confident and minimally overlapping.

5. **Mask and Keypoint Refinement** In tasks involving instance segmentation or pose estimation, additional postprocessing steps are required to refine the predicted masks and keypoints. For instance, the predicted masks can be refined by applying morphological operations such as dilation and erosion, which theoretically correspond to the process of cleaning up noisy binary predictions. Keypoint refinement involves smoothing and adjusting the predicted keypoints to better align with the underlying object structure, which can be understood as a form of structured prediction refinement, where the network corrects its initial estimates based on the

spatial and relational context of the keypoints.

6. **Panoptic Segmentation Fusion** For panoptic segmentation, the final postprocessing step involves fusing the outputs of instance and semantic segmentation branches to produce a coherent and unified segmentation map. Theoretically, this step addresses the challenge of integrating different types of predictions (i.e., instance-based and region-based) into a single, consistent output. This fusion process can be understood through the lens of multi-task learning, where the network must reconcile the outputs of different tasks to produce a holistic interpretation of the scene.

7. **Contextual Reasoning and Post-Hoc Analysis** Advanced postprocessing techniques may also involve contextual reasoning, where the model's predictions are adjusted based on the broader context of the image. For example, predictions can be refined by considering the spatial relationships between detected objects, their relative scales, and their positions within the scene. Theoretically, this can be viewed as a form of post-hoc analysis, where the network's predictions are subjected to additional scrutiny and refinement based on global scene information. This approach aligns with the broader concept of holistic scene understanding, where the goal is to produce a coherent and contextually-aware interpretation of the visual data.

3.2 Advantages of Using Faster R-CNN with Detectron2

Faster R-CNN, integrated with the Detectron2[15] framework, offers several key advantages for the specific task of detecting obstacles (such as seams, buttons, zippers) and identifying printable areas on upper garments. These benefits are particularly aligned with the requirements of high-precision garment customization applications.

1. **High Detection Accuracy:** The two-stage architecture of Faster R-CNN, consisting of the Region Proposal Network (RPN) and the final detection network, provides high accuracy in object detection. This is crucial for applications where precise identification and localization of small or intricate obstacles on garments are required [12].

2. **Versatility in Object Detection:** Faster R-CNN with Detectron2[15] supports multiple object detection tasks. It can be adapted to identify a wide range of obstacles and patterns across different garment types, thanks to its flexible configuration options, such as anchor boxes with various scales and aspect ratios. This versatility ensures robust performance across diverse garment styles and materials [**?**].

3. **Integration with Instance Segmentation:** By extending to Mask R-CNN within the Detectron2 framework, the model can not only detect obstacles but also perform instance-level segmentation. This capability is essential for applications requiring detailed mapping of printable areas, ensuring that printing is accurately placed around obstacles like seams or buttons [1].

4. **Efficient Feature Extraction:** The shared convolutional backbone in Faster R-CNN allows for efficient feature extraction, which is used both in generating region proposals and in final object detection. This shared computation enhances the model's efficiency, reducing redundancy and computational overhead [1].

5. **Customization and Scalability:** Detectron2 provides extensive customization options, including easy integration of custom datasets, model configurations, and training strategies. This flexibility is beneficial for tailoring the model to specific garment customization tasks, such as adjusting for different types of fabrics or obstacle characteristics [15].

3.3 Assumptions in the Context of Garment Customization

Deploying Faster R-CNN with Detectron2[15] for the specific task of detecting obstacles and identifying printable areas on garments involves several key assumptions, particularly when developing an annotated dataset independently:

1. **Creation of an Annotated Dataset:** It is assumed that you are creating a detailed annotated dataset, including ground truth labels for various obstacle types (e.g., buttons, zippers) and the delineation of printable areas. This dataset needs to be comprehensive and accurately labeled to train the model effectively. The process of annotation can be resource-intensive and requires a clear labeling schema to ensure consistency and accuracy [16].

2. **Diversity of Data:** While creating your dataset, it is crucial to include a diverse range of garment styles, colors, and materials. This diversity helps the model generalize better to different real-world scenarios, including variations in garment appearance, texture, and the presence of different obstacles [17].

3. **Consistency in Annotation Quality:** The quality and consistency of annotations are critical. Any variability in labeling, such as differences in how obstacles are marked or how printable areas are defined, can affect the model's training and accuracy. Ensuring consistent annotation standards is vital [18].

4. **Computational Resources:** Even with a custom dataset, significant computational resources are required for training the Faster R-CNN model, particularly due to the complex nature of the two-stage detection process and the size of the datasets typically involved. Access to high-performance GPUs is essential for efficient training and inference [1].

5. **Training and Validation Split:** It is assumed that the dataset will be appropriately split into training, validation, and test sets. This split is necessary to evaluate the model's performance accurately and to avoid overfitting. The creation of these splits should consider the distribution of different garment types and obstacles to ensure balanced and representative sets [19].

6. **Handling Occlusions and Complex Backgrounds:** The dataset should consider scenarios where obstacles may be partially occluded or garments may have complex backgrounds. This helps the model learn to distinguish between relevant features (obstacles and printable areas) and irrelevant ones, enhancing its robustness in real-world applications [18].

7. **Real-time Performance Adjustments:** If real-time performance is a requirement for your application, consider that Faster R-CNN, while accurate, is not inherently optimized for speed. Depending on the application, it may be necessary to explore model optimizations or consider alternative, faster architectures [20].

3.4 Mathematical Modelling

3.4.1 Pre-processing Steps:

The preprocessing steps involve resizing and padding images, adjusting bounding boxes, and updating annotations to ensure that the images and annotations are in a consistent format suitable for training the model. Furthermore, to improve the robustness of the model, various augmentations such as random flips, brightness adjustment, contrast adjustment, and random lighting could be performed.

1. **Anchor Box Generation:** Anchor boxes are predefined bounding boxes that serve as reference frames for object localization. Let $\mathbf{B}_k = (w_k, h_k, x_k, y_k)$ represent an anchor box with width w_k, height h_k, and center coordinates (x_k, y_k). The set of anchor boxes $\mathscr{B}$ is generated by varying w_k and h_k over a range of scales and aspect ratios:

$$\mathscr{B} = \{\mathbf{B}_k \mid w_k \in \mathscr{S}, h_k \in \mathscr{A} \cdot w_k, (x_k, y_k) \in \mathscr{P}\}, \tag{3.1}$$

   where $\mathscr{S}$ is the set of scales, $\mathscr{A}$ is the set of aspect ratios, and $\mathscr{P}$ is the set of anchor center positions.

2. **Resizing and Aspect Ratio Preservation:** Image resizing ensures all input images are of consistent size.

   Given an input image, we resize it while maintaining its aspect ratio and pad it to a target size. The scaling factor s is calculated as:

$$s = \min\left(\frac{W_{target}}{W_{img}}, \frac{H_{target}}{H_{img}}\right) \tag{3.2}$$

   where W_{target} and H_{target} are the target width and height, and W_{img} and H_{img} are the original width and height of the image. The new size (W_{new}, H_{new}) of the image is:

$$W_{new} = W_{img} \times s, \quad H_{new} = H_{img} \times s \tag{3.3}$$

   The image is then centered on a new canvas of size (W_{target}, H_{target}) with gray

padding. The position to paste the resized image is:

$$(x_{paste}, y_{paste}) = \left(\frac{W_{target} - W_{new}}{2}, \frac{H_{target} - H_{new}}{2} \right) \tag{3.4}$$

3. **Adjusting Bounding Boxes**

   Each bounding box (x, y, w, h) in the original image is adjusted according to the scaling factor s and the paste position (x_{paste}, y_{paste}):

$$x' = x \times s + x_{paste}, \quad y' = y \times s + y_{paste}, \quad w' = w \times s, \quad h' = h \times s \tag{3.5}$$

   The adjusted bounding box is (x', y', w', h').

4. **Data Normalization and Standardization:**

   Normalization scales pixel values $\mathbf{I}_{i,j}$ to a fixed range, typically [0, 1]:

$$\mathbf{I}_{\text{norm}} = \frac{\mathbf{I} - \min(\mathbf{I})}{\max(\mathbf{I}) - \min(\mathbf{I})}. \tag{3.6}$$

   Standardization adjusts the pixel values to have zero mean and unit variance:

$$\mathbf{I}_{\text{std}} = \frac{\mathbf{I} - \mu_{\mathbf{I}}}{\sigma_{\mathbf{I}}}, \tag{3.7}$$

5. **Data Augmentation** Data augmentation involves applying a series of transformations to input images $\mathbf{I}$ to generate a diverse set of training samples. Mathematically, if $\mathbf{I}$ is an input image, the augmented image $\mathbf{I}'$ can be represented as:

$$\mathbf{I}' = T(\mathbf{I}; \theta), \tag{3.8}$$

   where $T(\mathbf{I}; \theta)$ is a transformation function parameterized by θ. Common transformations include:

   - **Random Flip**

     A random flip is applied horizontally with probability p_h and vertically with

probability p_v. For horizontal flip:

$$x' = W_{img} - x - w \tag{3.9}$$

For vertical flip:

$$y' = H_{img} - y - h \tag{3.10}$$

- **Brightness Adjustment**

  Brightness adjustment changes the intensity of the pixels by a factor α_b in the range $[\alpha_{b_{min}}, \alpha_{b_{max}}]$. The new pixel value $I'(x,y)$ is:

$$I'(x,y) = \alpha_b \times I(x,y) \tag{3.11}$$

- **Contrast Adjustment**

  Contrast adjustment scales the difference between the pixel values and the mean pixel value μ_I by a factor α_c in the range $[\alpha_{c_{min}}, \alpha_{c_{max}}]$. The new pixel value $I'(x,y)$ is:

$$I'(x,y) = \alpha_c \times (I(x,y) - \mu_I) + \mu_I \tag{3.12}$$

- **Saturation Adjustment**

  Saturation adjustment modifies the intensity of the colors. If $I_{hsv}(x,y)$ represents the HSV value of the pixel, the new saturation S' is:

$$S' = \alpha_s \times S \tag{3.13}$$

  where α_s is in the range $[\alpha_{s_{min}}, \alpha_{s_{max}}]$.

- **Random Lighting**

  Random lighting adds a perturbation to the color channels. If $I_{rgb}(x,y)$ represents the RGB value of the pixel, the new pixel value is:

$$I'_{rgb}(x,y) = I_{rgb}(x,y) + \varepsilon \tag{3.14}$$

  where ε is a small random value.

3.4.2 Faster R-CNN Model:

We use the Faster R-CNN model from the Detectron2 library for garment classification, obstacle detection, and print area mapping.

1. **Region Proposal Network (RPN):** The RPN in Faster R-CNN and Mask R-CNN generates region proposals by sliding anchor boxes of different scales and aspect ratios over the feature maps produced by the backbone CNN. The objectness score for each anchor box is computed as:

$$p_i = \sigma(W_p^T \cdot \phi(I, a_i)) \tag{3.15}$$

where p_i is the objectness score for anchor box a_i, σ is the sigmoid function, W_p is a learned weight vector, and $\phi(I, a_i)$ is the feature vector extracted from the input image I at the spatial location corresponding to anchor box a_i.

2. **Bounding Box Regression:** The bounding box regression branch in Faster R-CNN and Mask R-CNN predicts offsets to refine the coordinates of the proposed bounding boxes. The predicted offsets $\hat{t}_i$ for anchor box a_i are computed as:

$$\hat{t}_i = W_t^T \cdot \phi(I, a_i) \tag{3.16}$$

where W_t is a learned weight vector, and $\phi(I, a_i)$ is the feature vector extracted from the input image I at the spatial location corresponding to anchor box a_i.

3. **Loss Function** The loss function for Faster R-CNN is a combination of classification loss and bounding box regression loss:

$$L = L_{cls} + L_{reg} \tag{3.17}$$

where L_{cls} is the cross-entropy loss for object classification, and L_{reg} is the smooth L1 loss for bounding box regression.

3.4.3 Post-processing Steps

The post-processing phase refines the model's predictions to enhance accuracy and remove redundant or irrelevant detections. These steps are essential for precise garment classification, obstacle detection, and printable area mapping.

1. **Non-Maximum Suppression (NMS):**

   NMS is employed to eliminate redundant bounding boxes by retaining only the most probable regions. Given a set of bounding boxes $B = \{b_1, b_2, \ldots, b_n\}$ with corresponding confidence scores $S = \{s_1, s_2, \ldots, s_n\}$, the process is as follows:

   - **Sorting:** Sort the bounding boxes by their confidence scores:

   $$\text{sort}(B, S) \quad \text{such that} \quad s_i \geq s_j \quad \text{for} \quad i < j \tag{3.18}$$

   - **Iterative Suppression:** For each bounding box b_i in the sorted list:

     (a) Compute the Intersection over Union (IoU) between b_i and each subsequent box b_j where $j > i$:

     $$\text{IoU}(b_i, b_j) = \frac{\text{Area}(b_i \cap b_j)}{\text{Area}(b_i \cup b_j)} \tag{3.19}$$

     (b) Suppress b_j if $\text{IoU}(b_i, b_j) >$ threshold.

   This method ensures that only the most confident and spatially distinct detections are preserved.

2. **Bounding Box Refinement:**

   After NMS, bounding box refinement is conducted to fine-tune the predicted box coordinates. Given an initial bounding box $b = (x, y, w, h)$, where (x, y) denotes the center and (w, h) the width and height:

   - **Predicted Offsets:** Let the predicted offsets be $(\Delta x, \Delta y, \Delta w, \Delta h)$. The refined bounding box b' is computed as:

   $$x' = x + w \cdot \Delta x, \quad y' = y + h \cdot \Delta y \tag{3.20}$$

$$w' = w \cdot e^{\Delta w}, \quad h' = h \cdot e^{\Delta h} \tag{3.21}$$

This step aligns the bounding boxes more closely with the actual object boundaries, improving detection accuracy.

3. **Score Thresholding:**

   To filter out low-confidence predictions, a score threshold τ is applied. Only predictions with confidence scores $s \geq \tau$ are retained:

$$B' = \{b \in B \mid s(b) \geq \tau\} \tag{3.22}$$

4. **Segmentation of Objects:**

   Objects are segmented into three primary categories: garment types, obstacle types, and printable part types. Let $G = \{g_1, g_2, \ldots, g_m\}$ represent the garment types, $O = \{o_1, o_2, \ldots, o_k\}$ the obstacles, and $P = \{p_1, p_2, \ldots, p_n\}$ the printable parts. The goal is to accurately segment each object in an image into its respective category, enabling garment type identification, obstacle detection, and printable area mapping.

5. **Obstacle Masking:**

   Define a masking function $M : G \times O \to \{0,1\}$, where $M(g_i, o_j) = 1$ if obstacle o_j affects garment g_i, and $M(g_i, o_j) = 0$ otherwise. This function is applied to exclude the regions occupied by obstacles from the printable area.

6. **Printable Area Determination:**

   For each garment g_i, the printable area $P(g_i)$ is computed as:

$$P(g_i) = \text{Area}(g_i) \times \prod_{o_j \in O} \left(1 - M(g_i, o_j) \cdot \text{Area}(o_j)\right) \tag{3.23}$$

   This equation subtracts the areas occupied by obstacles from the total garment area, yielding the actual printable region.

7. **Validation and Adjustment:**

- **IoU Validation:** The printable area mapping is validated by calculating the IoU between the predicted printable area $P(g_i)$ and a ground-truth printable area $P_{\text{gt}}(g_i)$:

$$\text{IoU}(P(g_i), P_{\text{gt}}(g_i)) = \frac{\text{Area}(P(g_i) \cap P_{\text{gt}}(g_i))}{\text{Area}(P(g_i) \cup P_{\text{gt}}(g_i))} \tag{3.24}$$

Adjustments are made if the IoU falls below a predefined threshold, ensuring that the detected printable areas closely match the ground truth.

3.5 System Block Diagram

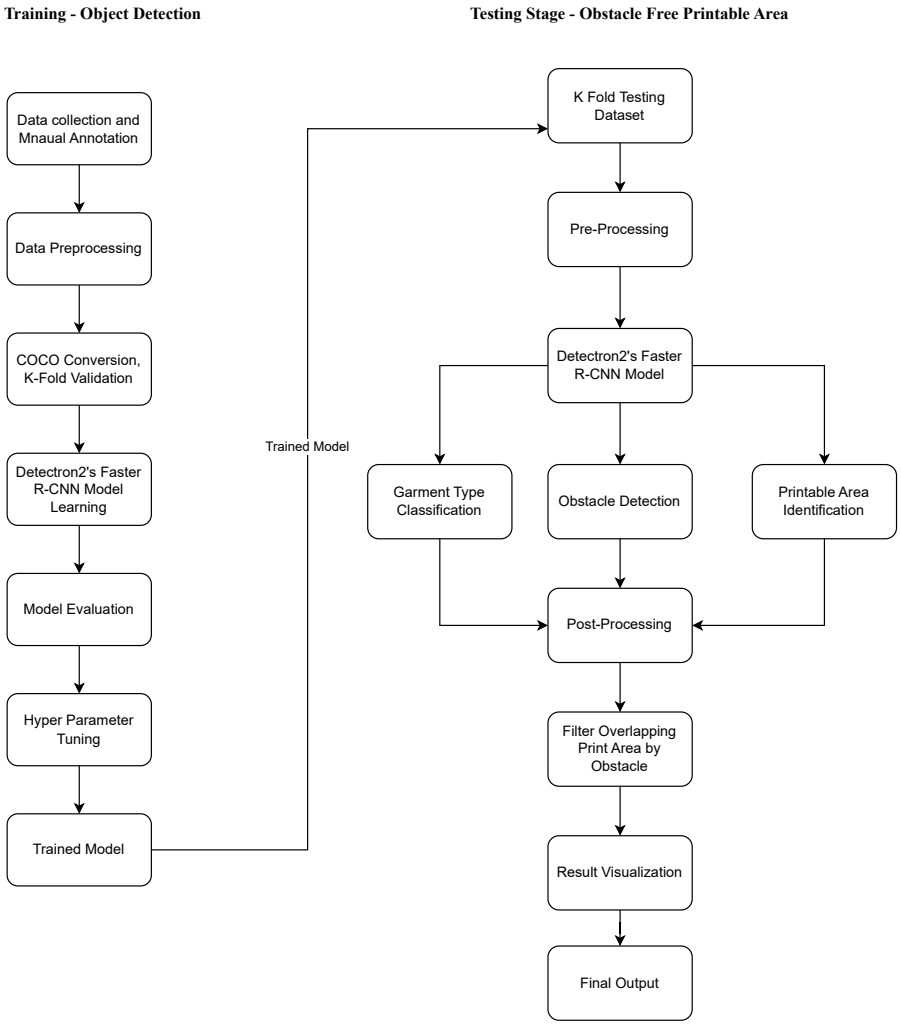


Figure 3.5: System block diagram for obstacle detection and print area mapping on garments using Faster R-CNN and Mask R-CNN.

The system block diagram shown in Figure 3.5 illustrates the end-to-end process of

obstacle detection and print area mapping on garments using Detectron2's Faster RCNN model. The process can be divided into two main phases: training and inference.

3.5.1 Training Phase:

1. **Input Data:** The training process begins with the collection of a diverse dataset of garment images. This dataset must include various garment types, such as t-shirts, jackets, and hoodies, and represent different obstacles that might affect printing, such as zippers, buttons, logos, seams, and pockets. The diversity of the dataset is crucial to ensure the model can generalize well across different scenarios.

   Once the images are collected, they undergo a manual annotation process. This involves using an annotation tool to draw bounding boxes around each relevant feature in the images, labeling these boxes with the corresponding categories (e.g., garment type, obstacle type). The annotated data will serve as the ground truth for training the object detection model, providing it with the necessary information to learn from the images.

2. **Data Preprocessing:** Before the images can be used to train the model, they must be preprocessed to ensure consistency and compatibility with the neural network. The first step in preprocessing is cleaning the dataset by removing any unnecessary or non-image files, such as DS Store files commonly found in macOS environments.

   Next, the images are resized and padded to a uniform target size. Resizing involves scaling the images so that their dimensions match a predefined size, typically one that the model expects during training. Care is taken to preserve the aspect ratio during this process to avoid distortion. Padding is applied if necessary, to maintain the aspect ratio while fitting the image into the target dimensions.

   After resizing and padding, the bounding boxes from the annotation step must be adjusted to align with the transformed images. This involves recalculating the coordinates of the bounding boxes to ensure they accurately reflect the location of objects in the resized images. Finally, the annotations themselves (usually stored in XML or JSON format) are updated to match the new image dimensions and bounding box coordinates.

3. **COCO Conversion and K-Fold Validation:** With the dataset prepared, the next step is to convert the annotations into the COCO (Common Objects in Context) format, which is a standard format widely used in object detection tasks. The COCO format organizes annotations into a structure that is compatible with many deep learning frameworks, including Detectron2, which will be used for training the Faster R-CNN model.

   Once the data is converted, the dataset is split into K subsets for cross-validation. In K-Fold cross-validation, the dataset is divided into K equal-sized folds, and the model is trained and validated K times, each time using a different fold as the validation set and the remaining K-1 folds as the training set. This technique provides a more robust evaluation of the model's performance by ensuring that every data point is used for both training and validation.

4. **Faster R-CNN Training:** The training phase itself involves using Detectron2's implementation of the Faster R-CNN model. Initially, the model is loaded with pre-trained COCO weights. These pre-trained weights provide a solid foundation, as they already capture general features from a large dataset, allowing the model to learn more specific garment-related features more effectively.

   The training process is iterative, involving multiple epochs where the model learns from the training data. During each epoch, the model makes predictions on the training images, compares these predictions to the ground truth annotations, and calculates the loss, which measures the difference between the predicted and actual values. The model then uses this loss to adjust its weights through backpropagation, gradually improving its accuracy.

5. **Model Evaluation:** To ensure the model performs optimally, it is evaluated periodically using metrics such as precision, recall, mean Average Precision (mAP), and ROC curves. These metrics provide insight into how well the model is distinguishing between different classes and how accurate its bounding box predictions are. Based on the evaluation results, hyperparameters such as learning rate, batch size, and the number of epochs may be adjusted to fine-tune the model's performance.

   After sufficient training and tuning, the model is finalized. The model that achieves

the best performance on the validation sets is selected as the final trained model, ready for deployment in the testing phase.

3.5.2 Inference Phase

1. **Data Preprocessing:** Before feeding the new images into the model for testing, the same preprocessing steps applied during training must be performed on the testing dataset. This ensures that the images are in the correct format and scale, allowing the model to process them consistently.

   As in the training phase, the images are resized and padded to the target dimensions, with bounding boxes adjusted accordingly. This preprocessing step is critical for maintaining the integrity of the model's predictions, as any discrepancies in image size or annotation format could lead to inaccurate results.

2. **Obstacle Detection (Faster R-CNN):** Once the testing data is preprocessed, the trained Faster R-CNN model is deployed to make predictions on these images. During inference, the model classifies the garment type in each image, identifying categories like t-shirts, jackets, or hoodies. Simultaneously, it detects obstacles such as zippers, buttons, logos, seams, and other features that might interfere with printing on the garment.

   In addition to classifying garment types and detecting obstacles, the model also identifies potential printable areas. These are regions on the garment where printing can occur without overlapping with any detected obstacles. The identification of these areas is crucial for applications like apparel design and manufacturing, where print placement needs to be precise and obstacle-free.

3. **Post-processing:** After the model makes its predictions, a series of post-processing steps are applied to refine the output. One key step is filtering overlapping printable areas by obstacles. This involves checking the detected printable regions against the detected obstacles and removing or adjusting any areas that overlap with obstacles. This ensures that the final printable areas are free of any obstructions, making them suitable for printing.

   Another important post-processing step is Non-Maximum Suppression (NMS). NMS is used to eliminate redundant bounding boxes that the model may have

predicted for the same object. For example, if the model predicts multiple overlapping boxes for a single logo, NMS will retain only the box with the highest confidence score and discard the others. This step is essential for ensuring the final predictions are clean and non-redundant.

Additionally, bounding box refinement is applied to fine-tune the predicted boxes. This step adjusts the boxes based on regression offsets, improving the accuracy of the box positions and sizes relative to the detected objects. Finally, score thresholding is used to filter out low-confidence predictions, ensuring that only reliable detections are included in the final output

4. **Output:** The final step in the testing/inference phase is visualizing and outputting the results. The model's predictions are rendered onto the images, showing the classified garment types, detected obstacles, and identified printable areas. These visualizations are essential for both validating the model's performance and for practical applications, where they can be used to guide printing processes.

   The final output is a set of refined predictions, ready for use in real-world applications such as garment printing or automated quality control in apparel manufacturing. The model's ability to accurately classify garments, detect obstacles, and identify printable areas makes it a valuable tool in these contexts

3.6 Instrumentation Requirements

The instrumentation for the project of mapping printable areas by detecting obstacles in upper garments has already been implemented, encompassing the necessary tools, software, and hardware. Below is an overview of the components used:

3.6.1 Hardware Tools:

1. **Personal Computer:** The project utilized an M1 Mac with 16 GB of RAM as the primary hardware platform.

   - **Device:** Apple M1 Mac with 16 GB of RAM.
   - **Purpose:** This device was employed for training the Faster R-CNN and Mask R-CNN models on the garment dataset and for performing inference during the garment customization process.

- **Access:** The device was readily available and fully utilized for the research.

2. **HD Camera:** An HD camera was used to capture additional garment images, supplementing the provided dataset as needed.

    - **Device:** A DSLR camera with at least 12 megapixels, capable of capturing images in RAW format.
    - **Purpose:** The camera was employed to capture high-quality images of various garment types, styles, and materials under controlled lighting conditions, enhancing the diversity and representativeness of the dataset.
    - **Access:** The camera was acquired and used as required during the data collection phase.

3.6.2 Software Tools:

1. **Deep Learning Framework:** A deep learning framework was essential for implementing the Faster R-CNN and Mask R-CNN models.

    - **Framework:** PyTorch, used within VSCode and Jupyter Notebook.
    - **Purpose:** The framework provided an isolated local environment for implementing the models, preprocessing data, and conducting model training and inference during the garment customization process.
    - **Access:** PyTorch was installed and configured on the M1 Mac, alongside VSCode and Jupyter Notebook, to facilitate development.

2. **Computer Vision Library:** OpenCV and Detectron2 were utilized for image processing and visualization tasks.

    - **Library:** OpenCV and Detectron2.
    - **Purpose:** OpenCV was used for tasks such as resizing, normalization, and data augmentation, as well as visualizing the results of obstacle detection and print area mapping. Detectron2 supported the model training and evaluation process.
    - **Access:** These libraries were integrated into the development environment on the M1 Mac.

3. **Annotation Tool:** A customized version of LabelImg was used for creating ground truth annotations.

    - **Tool:** Customized LabelImg.
    - **Purpose:** The tool was customized to meet the requirements for bounding box generation for garment object type classification, obstacle detection, and standard printable areas, adhering to the standards provided by Stahls.
    - **Access:** The customized tool was developed and deployed successfully, facilitating the annotation process.

3.7 Dataset Explanation

3.7.1 Relevancy of the Dataset:

The custom dataset is tailored for garment classification and obstacle detection, crucial tasks in the fashion and apparel industry. As companies increasingly rely on automation to streamline design, quality control, and production processes, the need for specialized datasets becomes evident. This dataset addresses the industry's challenges by enabling the development of models that can accurately identify garment types and detect obstacles like zippers, buttons, and seams, which are critical for printing and design workflows.

By focusing on specific upper-body garment types for males and incorporating a variety of obstacles, this dataset ensures that models trained on it will be robust and effective in real-world applications, such as automating quality checks and optimizing garment printing processes.

3.7.2 Exploration of the Dataset Contents

The dataset comprises images of various garment types, with a primary focus on:

- **T-shirts**
- **Shirts**
- **Polo Shirts**
- **Jackets**
- **Hoodies**

- **Sweatshirts**

The dataset may contains various obstacle in the above mentioned garments:

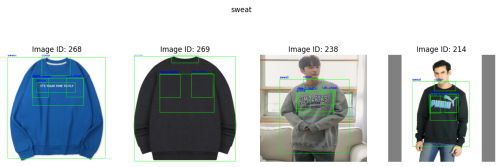
1. **Seam:** Lines where fabric pieces are sewn together.
2. **Zipper**: Metal or plastic fasteners used to join two edges of fabric.
3. **Pocket**: Small pouches sewn onto or into garments.
4. **Button**: Small fasteners, often circular, used for fastening garments.
5. **Prints**: Logo and different prints in the garments
6. **Text**: Text attached to garments, usually indicating brand or care instructions.
7. **Collar**: Collar of the garments.

Different garment types have different printing standards and some garment types may offer limited printing area and others offer more options. Following are the standards of Stahls Design Pattern [11] for locating and detecting the printable area of the above-mentioned types of garments:

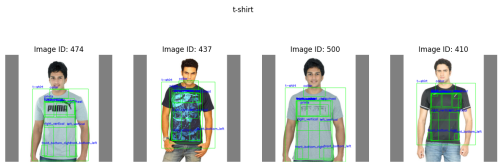
Garment wise standards:

1. **TShirt:** Left Chest, Right Chest, Center Chest, Across Chest,Left Vertical,Right Vertical, Front Bottom Left, Front Bottom Right, Full Front, Medium Front.
2. **Hoodie:** Front Center, Left Chest, Right Chest.
3. **Jacket:** Left Chest, Right Chest.
4. **Polo tShirt:** Left Chest, Right Chest.
5. **Sweat:** Front Center, Left Chest, Right Chest.
6. **Shirt:** Left Chest, Right Chest.

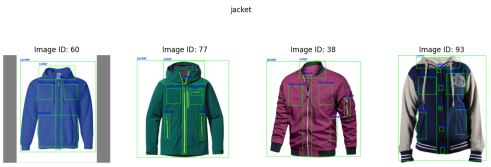
Each image in the dataset is annotated with bounding boxes to identify both the garment itself and any obstacles present. The annotations are detailed, including:



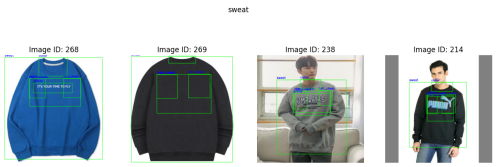
(a) Garment type: Hoodie



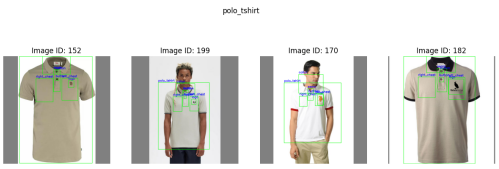
(b) Garment type: T-shirt



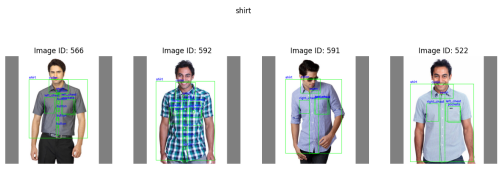
(c) Garment type: Jacket



(d) Garment type: Sweatshirt



(e) Garment type: Polo T-shirt



(f) Garment type: Polo Shirt

Figure 3.6: Dataset showing different garment types: Hoodie, T-shirt, Jacket, Sweatshirt, Polo T-shirt, and Polo Shirt.

- **Garment Type Labels:** Each image is categorized into one of the garment types mentioned above.
- **Obstacle Annotations:** These include bounding boxes for obstacles such as zippers, buttons, pockets, seams, and logos.
- **Printable Area Annotations:** Regions on the garment free from obstacles, suitable for placing designs or prints, are also annotated.

3.7.3 Dataset Preparation and Mechanism

In this research, The dataset collection process involved aggregating diverse sources to build a comprehensive dataset suitable for the task of mapping printable areas and detecting obstacles in upper garments. The dataset was meticulously curated to ensure that it encapsulated various garment types, styles, and conditions. The preparation mechanism, including the implementation of K-Fold cross-validation, involved several steps is detailed below:

1. **Data Sources:**

(a) **Fashion MNIST Dataset:**

- **Source:** Fashion MNIST dataset [21].
- **Garment Types:** T-shirts, shirts, polo shirts, jackets, hoodies, and sweatshirts.
- **Purpose:** This dataset provided a base set of high-quality images, serving as the primary dataset for garment type classification.

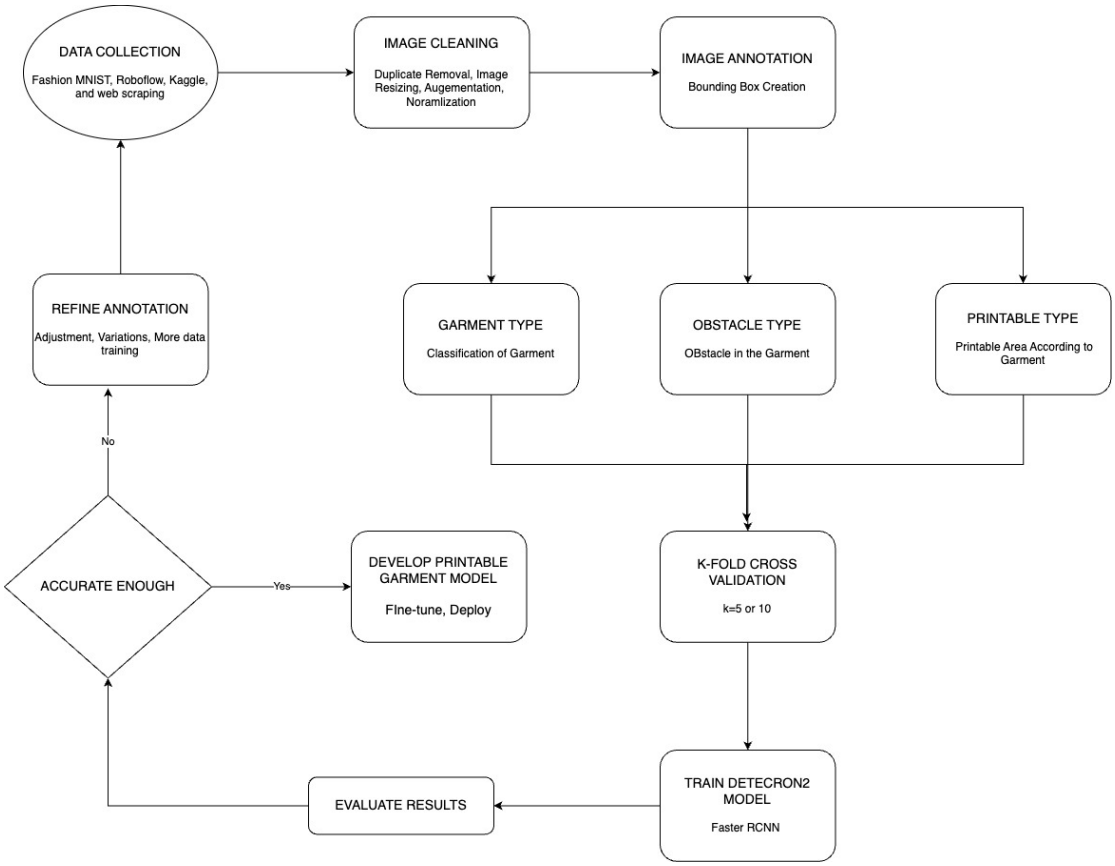


Figure 3.7: Flowchart depicting the dataset preparation and model training process

(b) **Roboflow Dataset:**

- **Source:** Roboflow Universe [22], [23], [24].
- **Supplemented Garments:** Focused on enhancing diversity in jacket and sweatshirt images.
- **Purpose:** The Roboflow dataset enriched the collection with additional variations in garment types.

(c) **Kaggle Dataset:**

- **Source:** Kaggle dataset [25].

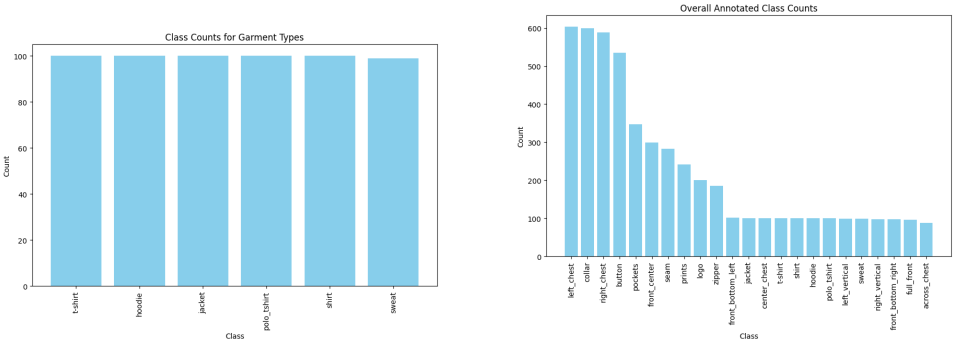
- **Purpose:** To introduce further variety in garment styles, colors, and designs, which included different fabric types and patterns.

(d) **Web Scraping:**

- **Source:** Collected from various online retail websites [26].
- **Purpose:** To supplement the dataset with a broader range of garment styles, ensuring comprehensive coverage of the fashion domain.

2. **Data Preparation and Preprocessing:**

(a) **Dataset Structure:** The dataset is organized into folders, each representing a different garment type. Each image is paired with a corresponding annotation file in JSON format. Metadata files provide additional context, such as garment size, color, and material, for more advanced models.



(a) Distribution of Garment Types (b) Overall Label Distribution

Figure 3.8: Garment Annotated Dataset distribution

(b) **Data Cleaning**

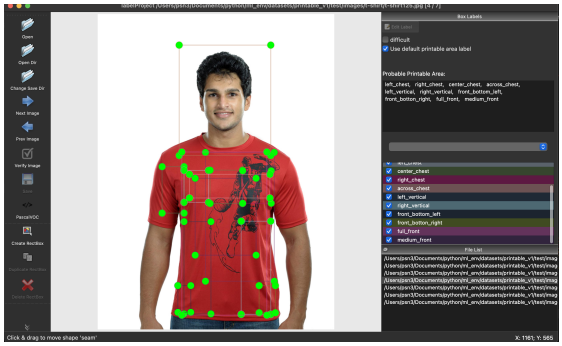
The collected data underwent rigorous cleaning to ensure quality:

- **Removing Duplicates and Low-Quality Data:** Duplicate images, as well as low-resolution or blurred images, were removed from the dataset.
- **Handling Missing Data:** We carefully addressed missing labels or incorrect annotations, ensuring that the dataset was complete and accurate for training.

(c) **Standardization Techniques:**

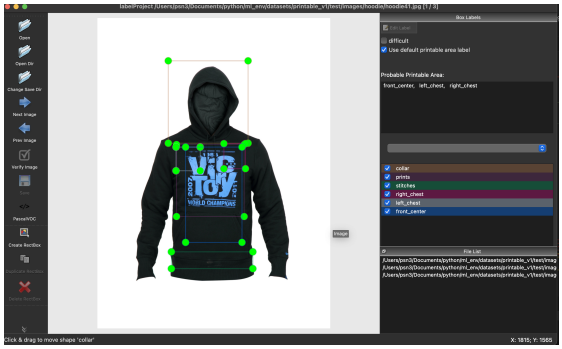
- **Resolution Standardization:** Uniform resizing of all images to 960*960 pixels was performed to ensure consistency across the dataset. This resolution was chosen to balance computational efficiency and the retention of important garment features.
- **Aspect Ratio Adjustment:** Images were either cropped or padded to maintain a consistent aspect ratio. This step was critical to preserving the spatial dimensions and proportions of garments, which is particularly important for obstacle detection and printable area mapping.
- **Color Normalization:** The color histograms of the images were normalized to reduce the variability caused by different lighting conditions. This standardization was crucial for improving model generalization and robustness.

(a) Original t-shirt Image



(b) Annotated t-shirt Image

(c) Original hoodie Image



(d) Annotated hoodie Image

Figure 3.9: Manual annotation of t-shirt and hoodie images

(d) **Annotation Procedure:**

The annotation process was carried out using a customized version of the LabelImg tool, which was tailored to meet the specific requirements of the project.

- **Annotation Tools:** Customized tools such as LabelImg were used to annotate the dataset with bounding boxes, segmentation masks, and classification labels.
- **Quality Control:** A review process was implemented where multiple annotators validated the labels, resolving disagreements and ensuring consistency in annotations.

Types of annotations performed on the diverse set of man upper garment types are :

- **Garment Type Labels:** Images were categorized into garment types (e.g., T-shirt, shirt, polo shirt, etc.), providing the model with explicit classes to learn and differentiate between.
- **Obstacle Annotations:** Detailed bounding boxes were drawn around obstacles such as zippers, buttons, pockets, seams, and logos. These annotations were essential for training the model to recognize and exclude these areas when mapping printable regions.
- **Printable Area Annotations:** Annotators meticulously identified areas of the garment free from obstacles, marking them as suitable for printing. This was done to train the model to correctly identify and suggest printable areas based on the absence of obstacles.

3. **K-Fold Cross-Validation Implementation Mechanism**

To ensure the robustness and generalizability of the model, a K-Fold Cross-Validation mechanism was employed. This method splits the dataset into K distinct subsets, allowing the model to be trained and validated across different partitions of the data.

(a) **Dataset Splitting:** The entire dataset was divided into K subsets (folds), where K was set to a standard value such as 5 or 10, depending on the dataset

size and computational resources available. Each fold served as a validation set exactly once, while the remaining K-1 folds were used for training.

(b) **Stratified Splitting:** To maintain a balanced representation of garment types and obstacles across all folds, a stratified splitting approach was used. This ensured that each fold had a similar distribution of different garment types and obstacles, preventing any bias towards specific classes during training.

(c) **Cross-Validation Training**

Each fold was used iteratively for training and validation:

i. **Iterative Training:** The model was trained on $k-1$ folds and validated on the remaining fold. This process was repeated k times, ensuring that each data point was used for both training and validation.

ii. **Averaging Results:** The performance metrics from all folds were averaged to obtain a reliable estimate of the model's effectiveness.

4. **Data Augmentation:**

To enhance the dataset, various data augmentation techniques were applied:

(a) **Synthetic Data Generation:**

Different image augmentation techniques were employed to create diverse data samples:

- **Image Augmentation Techniques:** Techniques such as rotation, scaling, color jitter, and noise addition were applied to generate variations of existing images. This increased the dataset's diversity and helped in reducing overfitting.
- **Balancing the Dataset:** Augmentation was also used to balance the dataset by generating additional samples for underrepresented classes.

(b) **Background Variation**

To further generalize the model, background variations were introduced:

- **Simulated Backgrounds:** Garments were placed in various simulated background settings to mimic real-world environments. This enhanced the model's ability to generalize across different contexts.

(c) **Final Dataset Preparation**

After K-Fold Cross-Validation and augmentation, the final dataset was prepared for training:

5. **Training, Validation, and Test Splits**

   The dataset was split into final training, validation, and test sets:

   - **Final Splitting:** The dataset was organized into training, validation, and test sets. The training set was used to train the model, the validation set for hyperparameter tuning, and the test set for final evaluation.
   - **Dataset Structuring:** The dataset was structured in a well-defined directory format, categorized by garment types, obstacles, and their corresponding annotations.

6. **Annotation Format**

   The annotations were converted into a standardized format:

   - **COCO Format:** All annotations were stored in COCO JSON format, which is compatible with most deep learning frameworks, facilitating easy training and evaluation.
   - **Metadata Integration:** Additional metadata, such as image resolution, aspect ratio, and color normalization parameters, was included to ensure consistency throughout the model development process.

7. **Evaluation and Feedback Loop**

   Finally, the model was evaluated, and an iterative feedback loop was implemented:

   (a) **Model Evaluation**

   The trained model was thoroughly evaluated:

   - **Validation on Holdout Set:** The model was validated on a holdout set not used during K-Fold Cross-Validation to ensure its performance on unseen data.

- **Error Analysis:** A detailed error analysis was conducted to identify common failure cases, such as misclassifications or missed obstacle detections.

(b) **Iterative Refinement**

Based on the evaluation, the dataset and model were refined:

- **Refining Annotations:** Annotations were refined in areas where the model underperformed. This involved re-annotating images or adding more detailed labels.
- **Augmentation Adjustments:** Adjustments were made to the augmentation parameters to optimize the balance between data diversity and relevance.

(c) **Deployment Preparation**

The final model was prepared for deployment:

i. **Model Selection**

The best-performing model was selected for deployment:

- **Choosing the Best Model:** After evaluation, the model with the highest performance metrics was selected for deployment.
- **Model Compression:** Techniques like quantization or pruning were employed to compress the model, making it suitable for deployment on resource-constrained devices.

ii. **Final Testing**

Before deployment, the model was subjected to final testing:

- **Test on Real-World Data:** The model was tested on a small set of real-world data to ensure it performs well outside the controlled dataset environment.
- **Continuous Monitoring Setup:** A monitoring system was set up to track the model's performance in the deployment environment, allowing for ongoing adjustments based on new data.

3.8 Description of Algorithms

3.8.1 Pre-Processing Algorithms

The major pre-processing algorithms needed for the obstacle detection and print area mapping project are:

1. **Image Resizing and Padding**

   Objective: Normalize image sizes to a consistent dimension while maintaining the aspect ratio to standardize the input data for the Faster R-CNN model.

   Mechanism:

   - **Input:** Original image with varying dimensions.
   - **Process:**
     (a) Calculate the aspect ratio of the image.
     (b) Resize the image to fit within a 960x960 pixel frame while maintaining the aspect ratio.
     (c) Pad the image with black pixels to achieve a final size of 960x960 pixels.
   - **Output:** Resized image with dimensions 960x960 pixels.

Algorithm 1 Image Resizing and Padding

Input: Image *image*, Target size $target\_size$ = (960, 960)

Output: Padded Image $padded\_image$

$original\_size \leftarrow image.size$ $aspect\_ratio \leftarrow original\_size[0]/original\_size[1]$

if $aspect\_ratio > 1$ **then**
  $new\_width \leftarrow target\_size[0]$ $new\_height \leftarrow new\_width/aspect\_ratio$

else
  $new\_height \leftarrow target\_size[1]$ $new\_width \leftarrow new\_height \times aspect\_ratio$

$resized\_image \leftarrow$ `Resize`*((image, (new_width, new_height))*

$padded\_image \leftarrow$ `New_Image`*("RGB", target_size)*

`Paste`*(padded_image, resized_image, ((target_size[0] - new_width) / 2, (target_size[1] - new_height)*

return $padded\_image$

2. **Data Augmentation**

Objective: Enhance the diversity of the dataset by applying transformations to existing images, such as rotations, flips, and color adjustments.

Mechanism:

- **Input:** Resized image.
- **Process:**
  (a) Apply random rotations (e.g., 0°, 90°, 180°, 270°).
  (b) Perform horizontal and vertical flips.
  (c) Adjust brightness, contrast, and saturation randomly within a defined range.
- **Output:** Augmented image with variations in orientation and color properties.

Algorithm 2 Data Augmentation

Input: Image *image*

Output: List of Augmented Images *augmented_images*

augmented_images ← [ ]

foreach *angle* ∈ [0, 90, 180, 270] **do**
    rotated_image ← `Rotate(`*(image, angle)*`)`
    augmented_images.append(*rotated_image*)
    augmented_images.append(`Mirror(`*(rotated_image)*`)`)
    augmented_images.append(`Flip(`*(rotated_image)*`)`)

foreach *img* ∈ *augmented_images* **do**
    img ← `Enhance_Brightness(`*(img, random.uniform(0.8, 1.2))*`)`
    img ← `Enhance_Contrast(`*(img, random.uniform(0.8, 1.2))*`)`
    img ← `Enhance_Color(`*(img, random.uniform(0.8, 1.2))*`)`

return *augmented_images*

3.8.2 Post-Processing Algorithms

After obtaining predictions from the trained Faster R-CNN model, the following post-processing steps are applied to refine the predictions, ensuring accurate detection of garment types, obstacles, and printable areas. This involves filtering, non-maximum suppression, bounding box refinement, and matching of predicted areas with actual requirements.

1. **Filtering and Overlap Removal:** This process is crucial for eliminating low-confidence predictions and resolving overlaps between identified obstacles and printable areas, ensuring that only valid printable regions are selected.

```
Algorithm: Filtering and Overlap Removal
1. For each prediction in outputs:
   a. Extract instances with their scores, classes, and bounding boxes.
   b. Identify highest scoring garment type.
   c. Filter instances by comparing scores to category-specific thresholds.
   d. Keep instances with scores above the threshold or the highest scoring
   e. Store the filtered instances.

2. For each filtered prediction:
   a. Identify indices of printable parts and obstacles.
   b. For each pair of printable parts and obstacles:
      i. Calculate IoU between them.
      ii. If IoU > 0.3:
          - Remove the overlapping printable part.
   c. Save the refined instances.
```

2. **Non-Maximum Suppression (NMS):** This step addresses redundancy by removing multiple bounding boxes that refer to the same object, thereby ensuring that only the most confident prediction is retained..

```
Algorithm: Non-Maximum Suppression (NMS)
1. For each prediction:
```

```
   a. Extract bounding boxes, scores, and classes.
   b. Apply NMS to filter out redundant bounding boxes.
   c. Store the filtered predictions.

2. For each unique class in predictions:
   a. Extract and concatenate bounding boxes and scores for that class.
   b. Apply NMS to the concatenated boxes.
   c. Store the filtered predictions.
```

3. **Bounding Box Refinement:** This process refines the location and size of the detected bounding boxes to improve accuracy.

```
Algorithm: Bounding Box Refinement
1. For each bounding box:
   a. Adjust coordinates to minimize the difference with ground-truth boxes
   b. Save the refined bounding boxes.
```

4. **Annotation Parsing and Label Encoding** This algorithm parses the annotations and converts them into a numerical format that is suitable for model training and evaluation.

```
Algorithm: Annotation Parsing and Label Encoding
1. For each annotation file:
   a. Parse the annotation data to extract object classes and bounding boxe
   b. Encode the object classes as numerical labels.
   c. Save the parsed and encoded annotations.
```

5. **IoU Computation and Matching:** This algorithm computes the Intersection over Union (IoU) between the predicted bounding boxes and ground-truth boxes to evaluate the accuracy of the predictions.

```
Algorithm: IoU Computation and Matching
1. For each predicted box:
   a. For each ground truth box:
```

i. Calculate IoU.

ii. If IoU > threshold:

- Consider it a match.

b. Record the matched predictions and ground truths.

3.9 Post-Processing Algorithms

In this section, we detail the major post-processing algorithms used in the project. These algorithms include Non-Maximum Suppression (NMS), Bounding Box Refinement, Score Thresholding, Object Segmentation, Obstacle Masking, Printable Area Determination, Validation and Adjustment, Annotation Parsing, and Label Encoding. Each algorithm's working mechanism is explained, accompanied by pseudo-code and algorithmic steps for clarity.

3.9.1 Non-Maximum Suppression (NMS)

Non-Maximum Suppression (NMS) is employed to eliminate redundant bounding boxes that represent the same object, keeping only the one with the highest confidence score. The algorithm operates as follows:

Algorithm 3 Non-Maximum Suppression (NMS)

Input: List of bounding boxes B, confidence scores S, IoU threshold t
Sort bounding boxes B based on scores S in descending order
Initialize an empty list K for keeping selected boxes **while** *there are boxes in* B **do**
    Select the box b with the highest score from B
Add b to K
Remove b from B **for** *each box* b_i *in* B **do**
    Compute IoU between b and b_i **if** $IoU(b, b_i) > t$ **then**
    Remove b_i from B
Output: List of selected bounding boxes K

3.9.2 Bounding Box Refinement

Bounding Box Refinement aims to improve the accuracy of predicted bounding boxes by adjusting their position and size using predicted offsets. The process is outlined below:

Algorithm 4 Bounding Box Refinement

1: **Input:** Initial bounding box coordinates (x, y, w, h), predicted offsets $(\Delta x, \Delta y, \Delta w, \Delta h)$

2: Compute the new center coordinates:

$$x' = x + w \cdot \Delta x,$$
$$y' = y + h \cdot \Delta y$$

3: Compute the new width and height:

$$w' = w \cdot \exp(\Delta w),$$
$$h' = h \cdot \exp(\Delta h)$$

4: **Output:** Refined bounding box coordinates (x', y', w', h')

3.9.3 Score Thresholding

Score Thresholding filters out predictions with low confidence scores, retaining only the most reliable detections. The algorithm is as follows:

Algorithm 5 Score Thresholding

1: **Input:** Predictions P, confidence scores S, threshold t

2: Initialize an empty list F for filtered predictions **for** *each prediction* p_i *in* P **do**
$S_i \geq t$

3: Add p_i to F

4:

5:

6: **Output:** Filtered predictions F

3.9.4 Object Segmentation

Object Segmentation categorizes objects into garment types, obstacle types, and printable part types based on model predictions. The categorization process is described below:

Algorithm 6 Object Segmentation

Input: Predictions P, labels L
Initialize empty lists G, O, P_p for garments, obstacles, and printable parts, respectively **for** *each prediction* p_i *in* P **do**
    $L_i \in$ Garment Labels
Add p_i to G
$L_i \in$ Obstacle Labels
Add p_i to O
$L_i \in$ Printable Part Labels
Add p_i to P_p

Output: Categorized objects G, O, P_p

3.9.5 Obstacle Masking

Obstacle Masking excludes regions affected by obstacles from being considered as printable areas. The masking procedure is as follows:

Algorithm 7 Obstacle Masking

Input: Garments G, obstacles O, masking function M
Initialize an empty list M_r for masked regions **for** *each garment* g_i *in* G **do**
    Set initial mask $m_i = 1$ **for** *each obstacle* o_j *in* O **do**
    $M(g_i, o_j)$ is true
$m_i \leftarrow m_i - \text{Area}(o_j)$
Append $g_i \cdot m_i$ to M_r

Output: Masked regions M_r

3.9.6 Printable Area Determination

This step calculates the actual printable area on a garment after accounting for obstacle regions:

Algorithm 8 Printable Area Determination

1: **Input:** Garments G, obstacles O

2: Initialize an empty list A_p for printable areas **for** *each garment* g_i *in* G **do**

3:

Compute the printable area:

$$A_{p_i} = \text{Area}(g_i) - \sum_{o_j \in O} \text{Area}(o_j)$$

4: Append A_{p_i} to A_p

5:

6: **Output:** Printable areas A_p

3.9.7 Validation and Adjustment

The Validation and Adjustment step ensures the accuracy of printable areas by validating against ground truth and adjusting if necessary:

Algorithm 9 Validation and Adjustment

1: **Input:** Predicted areas A_p, ground truth areas A_g, IoU threshold t

2: Initialize an empty list A_v for validated areas **for** *each predicted area* a_i *in* A_p **do**

3:

Compute IoU between a_i and $A_g[i]$ **if** $IoU(a_i, A_g[i]) < t$ **then**

4:

Adjust a_i based on $A_g[i]$

5:

6: Append a_i to A_v

7:

8: **Output:** Validated and adjusted printable areas A_v

3.9.8 Annotation Parsing

Annotation Parsing processes the raw data annotations into a structured format for further processing:

Algorithm 10 Annotation Parsing

Input: Raw annotations R

Initialize an empty list A for parsed annotations **for** *each annotation* r_i *in* R **do**

    Parse bounding box coordinates (x, y, w, h)

Parse object class label L

Append parsed data (x, y, w, h, L) to A

Output: Parsed annotations A

3.9.9 Label Encoding

Label Encoding transforms categorical labels into numerical format for model input:

Algorithm 11 Label Encoding

Input: Labels L, label map M

Initialize an empty list E for encoded labels **for** *each label* l_i *in* L **do**

    Encode l_i using the label map M: $e_i = M[l_i]$

Append e_i to E

Output: Encoded labels E

3.10 Flowcharts and Pseudo-Code

The following subsections provide visual representations and pseudo-code to support the explanation of the post-processing steps mentioned above.

3.10.1 Flowchart for Post-Processing Workflow

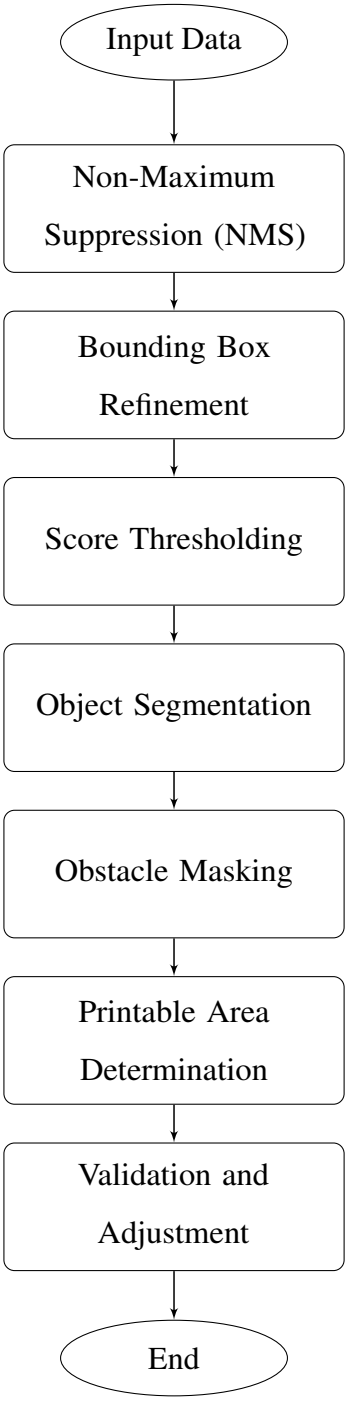


Figure 3.10: Flowchart of the Post-Processing Workflow

3.10.2 Pseudo-Code for Annotation Parsing and Label Encoding

Below is the pseudo-code representing the annotation parsing and label encoding processes. These algorithms ensure that the data is prepared correctly for model training and prediction.

Algorithm 12 Annotation Parsing

Input: Raw annotations R

Initialize an empty list A for parsed annotations **for** *each annotation* r_i *in* R **do**

    Parse bounding box coordinates (x, y, w, h)

Parse object class label L

Append parsed data (x, y, w, h, L) to A

Output: Parsed annotations A

Algorithm 13 Label Encoding

Input: Labels L, label map M

Initialize an empty list E for encoded labels **for** *each label* l_i *in* L **do**

    Encode l_i using the label map M: $e_i = M[l_i]$

Append e_i to E

Output: Encoded labels E

3.11 Elaboration of Working Principle

3.11.1 Preprocessing of Raw Input Data

1. **Raw Data to ML-Ready Data:** Raw input data consists of images of upper garments, which need to be preprocessed to make them suitable for machine learning models. The preprocessing steps include:

    - **Image Resizing and Padding:** Raw images are resized to a consistent dimension to standardize the input for the ML model. The target size is typically set to 960×960 pixels. Padding is added to maintain the aspect ratio of the original image.
    - **Data Augmentation:** To increase the diversity of the training dataset and improve the robustness of the model, data augmentation techniques such as rotation, flipping, scaling, and color jittering are applied.
    - **Annotation for Object Detection:** Annotations are necessary to inform the model of the objects to detect. This involves labeling images with bounding

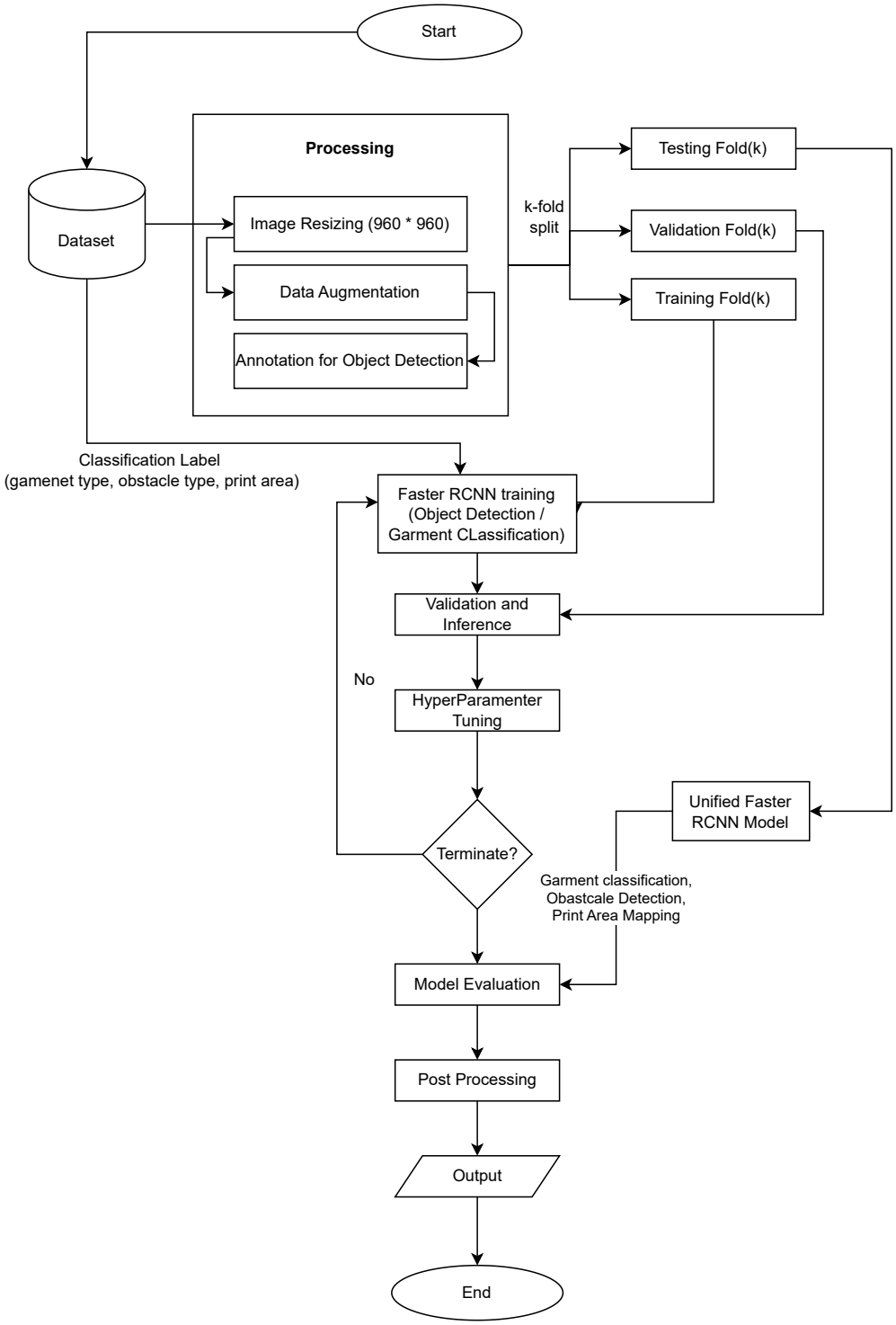


Figure 3.11: Flow Chart for obstacle detection and print area mapping on garments using Faster R-CNN and Mask R-CNN.

boxes and class labels for each object. The annotations are typically saved in COCO format. When padding is added during preprocessing, the annotations

must be updated to ensure consistency. The bounding box coordinates are adjusted based on the padding offsets.

- **Normalization:** Detectron2 normalizes images using the mean and standard deviation values of the ImageNet dataset by default. The mean values are [103.530, 116.280, 123.675], and the standard deviation values are [1.0, 1.0, 1.0]. However, custom normalization can be specified in the configuration.

  Example of setting normalization in Detectron2:

```
from detectron2.config import get_cfg
from detectron2.engine import DefaultPredictor

cfg = get_cfg()
cfg.merge_from_file("path/to/your/config.yaml")
cfg.MODEL.WEIGHTS = "path/to/your/model.pth"
cfg.INPUT.FORMAT = "BGR"
cfg.INPUT.MEAN = [103.530, 116.280, 123.675]
cfg.INPUT.STD = [1.0, 1.0, 1.0]

predictor = DefaultPredictor(cfg)
```

  If custom normalization is required before feeding the images to the model, it can be done manually:

```
import cv2
import numpy as np

mean = [103.530, 116.280, 123.675]
std = [1.0, 1.0, 1.0]

def normalize_image(image):
    image = image.astype(np.float32)
    image = (image - mean) / std
    return image
```

```
image = cv2.imread('path_to_your_image.jpg')
normalized_image = normalize_image(image)
```

.

- **K-Fold Split:** The dataset is split into k-folds to perform cross-validation. This helps in evaluating the model's performance more reliably.

3.11.2 Model Manipulation Through the Stages of Faster R-CNN with FPN Backbone

1. **Data Input and Preprocessing**

   - **Input Data: Images:** The input images are resized to 960 × 960 960×960 pixels and normalized. This ensures consistency and helps the model learn more effectively.
   - **Annotations:** The annotations (bounding boxes and class labels) are also scaled accordingly to match the resized images.

2. **Backbone (ResNet + FPN)**

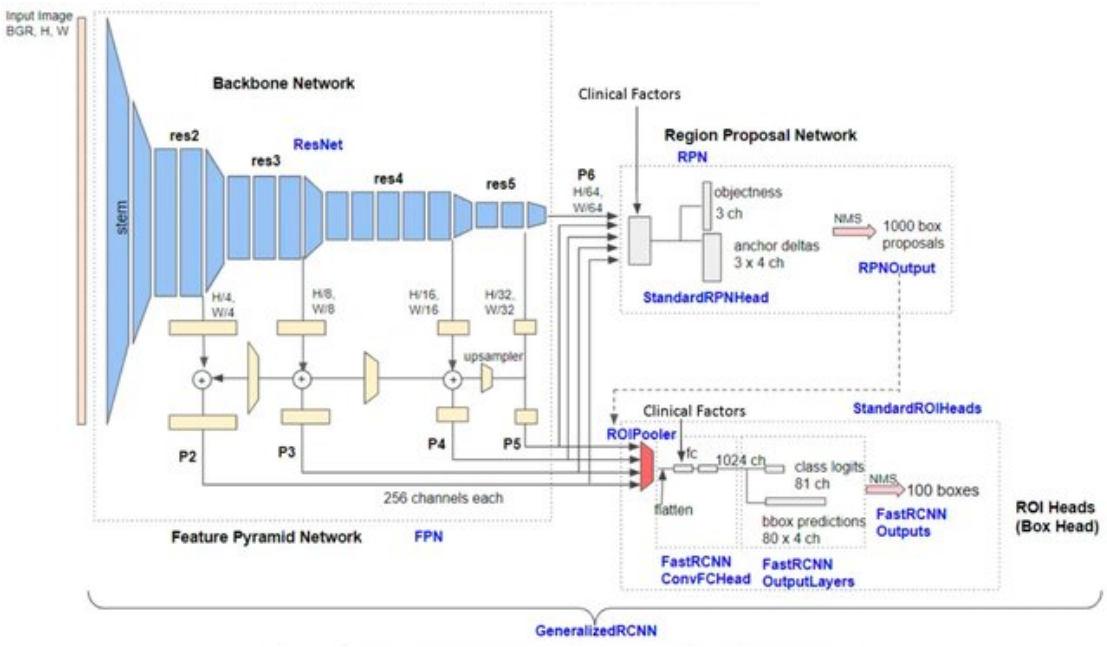


Figure 3.12: Detailed architecture of Base-RCNN-FPN [2]

- **ResNet Stages:**
  (a) **Initial Convolution and Pooling:**
    – The input image ($960 \times 960 \times 3$ 960×960×3) is passed through a 7×7 7×7 convolutional layer with a stride of 2, producing a feature map of size $480 \times 480 \times 64$ 480×480×64.
    – This is followed by a 3×3 3×3 max pooling layer with a stride of 2, reducing the feature map size to $240 \times 240 \times 64$ 240×240×64.
  (b) **Res2 Stage:**
    – The $240 \times 240 \times 64$ 240×240×64 feature map is processed through a series of bottleneck blocks, resulting in a feature map of size $240 \times 240 \times 256$ 240×240×256.
  (c) **Res3 Stage:**
    – The $240 \times 240 \times 256$ 240×240×256 feature map is processed through additional bottleneck blocks, producing a feature map of size $120 \times 120 \times 512$ 120×120×512.
  (d) **Res4 Stage:**
    – The $120 \times 120 \times 512$ 120×120×512 feature map is processed further, yielding a $60 \times 60 \times 1024$ 60×60×1024 feature map.
  (e) **Res5 Stage:**
    – Finally, the $60 \times 60 \times 1024$ 60×60×1024 feature map is processed, resulting in a $30 \times 30 \times 2048$ 30×30×2048 feature map.
- **Feature Pyramid Network (FPN):**The FPN builds a pyramid of feature maps from different ResNet stages:
  (a) **P2:** Derived from Res2 ($240 \times 240 \times 256$ 240×240×256).
  (b) **P3:** Derived from Res3 ($120 \times 120 \times 512$ 120×120×512).
  (c) **P4:** Derived from Res4 ($60 \times 60 \times 1024$ 60×60×1024).
  (d) **P5:** Derived from Res5 ($30 \times 30 \times 2048$ 30×30×2048).
  (e) **P6:** Obtained by max pooling P5 ($15 \times 15 \times 256$ 15×15×256).

  Each FPN level combines features from its corresponding ResNet stage and the previous FPN level through lateral and output connections.

3. **Region Proposal Network (RPN)**

   - **RPN Head:**

     (a) **Feature Extraction:** Each FPN feature map is processed by a 3×3 3×3 convolutional layer, maintaining the spatial dimensions but reducing the number of channels.

     (b) **Objectness Logits:** A 1×1 1×1 convolutional layer predicts objectness scores for each anchor, indicating the likelihood of an object being present.

     (c) **Anchor Deltas:** Another 1×1 1×1 convolutional layer predicts bounding box adjustments for each anchor.

   For example, consider P2 ($240 \times 240 \times 256$ 240×240×256):

   (a) **Objectness Logits:** $240 \times 240 \times 3$ 240×240×3.

   (b) **Anchor Deltas:** $240 \times 240 \times 12$ 240×240×12.

4. **ROI Heads**

   - **ROI Pooling and Box Head:**

     (a) **ROI Pooling:** The region proposals from the RPN are pooled to a fixed size (e.g., 7×7 7×7) using ROIAlign.

     (b) **Fully Connected Layers:** The pooled feature maps are flattened and passed through fully connected layers to produce feature vectors.

     (c) **Box Predictor:**

       – **Class Scores:** Predicts the class probabilities for each region.

       – **Bounding Box Deltas:** Refines the bounding box coordinates.

3.11.3 Post-Processing of Model Output

1. **Processing Model Output** The model outputs classification labels for detected objects. Post-processing involves mapping these labels to the respective garment types, obstacle types, and print areas.

   - **Non-Maximum Suppression (NMS):** The bounding boxes predicted by the model are refined using techniques such as non-maximum suppression

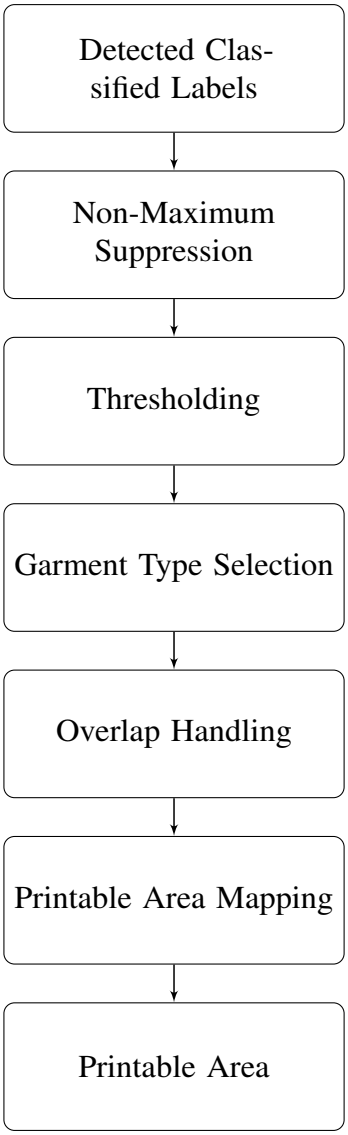


Figure 3.13: Post-Processing Workflow

(NMS) to remove duplicate detections and keep only the most confident ones.

- **Thresholding:** Applies a confidence threshold to filter out low-confidence detections. Different thresholds are used for different categories:
  (a) Garment types: 0.5
  (b) Obstacles: 0.4
  (c) Printable areas: 0.3
- **Garment Type Selection:** For each image, only the highest-scoring garment type detection is retained, ensuring that only one garment type is identified per image.

- **Overlap Handling:** The system checks for overlaps between printable areas and obstacles:
  (a) If a printable area overlaps more than 30% with any obstacle, it is removed.
  (b) This ensures that suggested print areas do not coincide with obstacles on the garment.
- **Printable Area Mapping:** After removing overlapping areas, the remaining printable areas are identified as suitable for logo placement.

2. **Non-Maximum Suppression** Assume two bounding boxes with a high Intersection over Union (IoU) of 0.7. NMS keeps the box with the higher confidence score and removes the other:

$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}} = 0.7$$

3.11.4 Sample Calculations

Let's illustrate the process with a concrete example for the first few stages:

Image Resize

Consider an input image resized to 960×960 pixels with an aspect ratio of 1.5 (i.e., width is 1.5 times the height).

$$\begin{aligned}
\text{original_size} &= (1440, 960) \\
\text{aspect_ratio} &= \frac{1440}{960} = 1.5 \\
\text{new_width} &= 960 \\
\text{new_height} &= \frac{960}{1.5} = 640 \\
\text{padded_image} &= \text{Image.new}(\text{"}RGB\text{"}, (960, 960)) \\
\text{offset} &= \left(\frac{960 - 960}{2}, \frac{960 - 640}{2} \right) = (0, 160)
\end{aligned}$$

The resized image will be centered in a 960×960 canvas with padding added to the top

and bottom.

Initial Convolution and Pooling

Input Image:

$$\text{image\_size} = 960 \times 960 \times 3$$

Convolution (7x7, stride 2):

$$\text{Output height/width} = \left(\frac{960 - 7 + 2 \times 3}{2}\right) + 1 = 480.$$

$$\text{Output Feature Map} = 480 \times 480 \times 64.$$

Max Pooling (3x3, stride 2):

$$\text{Output height/width} = \left(\frac{480 - 3 + 2 \times 1}{2}\right) + 1 = 240.$$

$$\text{Output Feature Map} = 240 \times 240 \times 64.$$

Res2 Stage

Input Feature Map: 240 x 240 x 64

Bottleneck Block:

- Assuming a bottleneck block with dimensions: 1×1×256, 3×3×64, 1×1×256
- **Calculation for the first convolution (1x1x256):**

$$\text{Output height/width} = 240 \quad (\text{no change}).$$

$$\text{Output channels} = 256.$$

$$\text{Output Feature Map} = 240 \times 240 \times 256.$$

RPN Objectness and Bounding Box Prediction

Input Feature Map (P2):

$$\text{Input Image} = 240 \times 240 \times 256$$

Objectness Logits (1x1 convolution):

$$\begin{aligned} \text{Output height/width} &= 240 \quad (\text{no change}). \\ \text{Output channels} &= 3 \quad (\text{number of anchors}). \\ \text{Output Feature Map} &= 240 \times 240 \times 3. \end{aligned}$$

Bounding Box Deltas (1x1 convolution):

$$\begin{aligned} \text{Output height/width} &= 240 \quad (\text{no change}). \\ \text{Output channels} &= 12 \quad (4 \text{ coordinates per anchor} \times 3 \text{ anchors}). \\ \text{Output Feature Map} &= 240 \times 240 \times 12. \end{aligned}$$

3.12 Verification and Validation

Verification and validation are crucial stages in ensuring the accuracy and reliability of the developed models for obstacle detection in upper garments and mapping printable areas. This section discusses the relevance of the chosen metrics in evaluating the output of the models.

3.12.1 Chosen Metrics and Their Relevance

To evaluate the performance of the Detectron2 Faster R-CNN model, we utilize a combination of quantitative metrics and visual evaluation tools. These include Precision, Recall, F1-Score, Mean Average Precision (mAP), Intersection over Union (IoU), Average Precision (AP), Confusion Matrix, ROC Curve, Loss Curve, and Learning Curve.

3.12.2 Relevance of Chosen Metrics

The following metrics have been selected to judge the performance of the obstacle detection and printable area mapping models:

- **Accuracy**: Measures the overall correctness of the model's predictions. In the context of obstacle detection, accuracy indicates the proportion of correctly identified obstacles. For printable area mapping, it signifies the accuracy of mapping printable areas.

- **Precision**: Precision is relevant in evaluating the model's ability to avoid false positives. In obstacle detection, precision indicates the proportion of correctly identified obstacles out of all predicted obstacles. Similarly, in printable area mapping, precision reflects the accuracy of identifying printable areas without including non-printable regions.

- **Recall**: Recall is essential for assessing the model's capability to detect all relevant instances. In obstacle detection, recall measures the proportion of correctly identified obstacles out of all actual obstacles present in the dataset. For printable area mapping, recall indicates the ability to identify all printable areas.

- **Intersection over Union (IoU)**: IoU is particularly relevant for evaluating the accuracy of bounding box predictions in obstacle detection. It measures the overlap between predicted and ground truth bounding boxes, providing insight into the localization accuracy of detected obstacles.

- **Mean Average Precision (mAP)**: mAP offers a comprehensive evaluation of the model's performance across multiple classes. It considers both precision and recall, providing a balanced assessment of the model's ability to detect obstacles and map printable areas accurately.

  Confusion Matrix: The confusion matrix is particularly useful in multi-class object detection tasks to understand the model's performance on each class, highlighting where it confuses one class for another.

These metrics are relevant because they provide a quantitative way to evaluate the model's detection and segmentation capabilities, highlighting both the strengths and weaknesses of the model in different scenarios.

3.12.3 Definitions and Formulas

- **Intersection over Union (IoU)**: The IoU metric quantifies the degree of overlap between predicted and ground truth bounding boxes. It is computed by dividing the area of overlap by the area of their union.

$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}} \tag{3.25}$$

- **Precision**: Precision measures the accuracy of positive predictions, representing the ratio of true positives to the sum of true positives and false positives

$$\text{Precision} = \frac{TP}{TP + FP} \tag{3.26}$$

where TP is the number of true positives and FP is the number of false positives.

- **Recall**: Recall evaluates the completeness of positive predictions, denoting the ratio of true positives to the sum of true positives and false negatives.

$$\text{Recall} = \frac{TP}{TP + FN} \tag{3.27}$$

where FN is the number of false negatives.

- **F1-Score**: The F1-Score harmonizes precision and recall into a single metric, offering a balanced assessment of model performance.

$$\text{F1-Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3.28}$$

- **Mean Average Precision (mAP)**: Mean Average Precision provides an aggregate measure of precision across all classes, offering a comprehensive evaluation of model effectiveness.

$$\text{mAP} = \frac{1}{N} \sum_{i=1}^{N} AP_i \tag{3.29}$$

where AP_i is the average precision for class i and N is the number of classes.

- **Confusion Matrix**: A confusion matrix provides a detailed breakdown of the model's predictions versus the actual labels, categorizing outcomes as True Positive

(TP), False Positive (FP), True Negative (TN), and False Negative (FN)

$$\text{Confusion Matrix} = \begin{pmatrix} \text{TP} & \text{FP} \\ \text{FN} & \text{TN} \end{pmatrix} \tag{3.30}$$

4 RESULTS

This chapter presents the results of our experiments aimed at classifying garment types and detecting obstacles to accurately identify printable areas using the Faster R-CNN model. The primary objective was to develop a robust method for detecting and mapping printable regions on male upper body garments by leveraging deep learning techniques. Our methodology involved experimenting with different dataset sizes, batch sizes, and training epochs to fine-tune the model and optimize its performance. We evaluated the model using key metrics such as Mean Average Precision (mAP) and Intersection over Union (IoU) to assess its effectiveness in accurately detecting obstacles and identifying unobstructed printable areas.

We conducted four main experiments, supplemented by multiple early stopping trials, to analyze the impact of various configurations and identify the optimal setup. The experiments focused on six classes of obstacles commonly found on garments, such as zippers, buttons, and pockets, which are critical to determining whether an area is suitable for printing.

The results of four key experiments, which involved different configurations of dataset sizes, batch sizes, iterations, and learning rates, are summarized in Table 4.1. These experiments provided insights into how different factors influenced the model's ability to classify garment types and detect obstacles, guiding us toward the most effective strategies for achieving our research objectives.

Table 4.1: Summary of Major Experiments

Exp.	**Dataset**	**Batch**	**Iterations**	**k-fold**	**LR**
1	120 (20/class)	2	1000 iters	-	0.025
2	120 (20/class)	4	1000 iters	-	0.025
3	1200 (100/class)	2	700 iters/fold	5	0.025
4	1200 (100/class)	2	3000 iters/fold	10	0.025

Each experiment was designed to understand how different configurations affected the model's ability to accurately classify garment types and detect obstacles, thereby determining the best strategies for detecting printable areas. The following sections

provide a detailed analysis of each experiment's outcomes, focusing on the impact of dataset size, batch size, iterations, and learning rate on model performance.

4.1 Initial Setup and Early Findings

The preliminary experiments were designed to establish a performance baseline for garment classification, obstacle detection, and printable area localization using the Faster R-CNN implementation provided by Detectron2. The experimental dataset comprised a subset of 20 images per garment category, meticulously annotated for obstacles and printable regions. Furthermore, the dataset was split into training, validation, and testing sets in a 70:15:15 ratio.

4.1.1 Experiment One with 1000 Iterations:

Table 4.2: Experiment One Hyperparameter Configuration

Hyperparameter	**Value**
Backbone	R_50_FPN_3x.yaml
Model weights	R_50_FPN_3x.yaml (Model Zoo)
No. of Workers	2
Batch Size	2
Base LR	0.00025
Max. Itr.	1000
ROI Head per Img	128
ROI Head Class	23

Best-Case Scenario

- **AP_{50} Performance:** The highest performance in Experiment 2 was observed with AP_{50} (IoU=0.50), where the model achieved a precision of 42.317%. This indicates that the model was relatively accurate in detecting and classifying certain garment features with at least 50% overlap between the predicted bounding boxes and the ground truth.
- **Category Performance - Collar:** The collar category demonstrated a relatively strong performance with an Average Precision (AP) of 31.476%, indicating that

the model was more adept at identifying and localizing collars in this run.

Worst-Case Scenario

- **AP_{75} Performance:** The AP_{75} (IoU=0.75) metric was significantly lower, at 6.830%, indicating that the model struggled with higher IoU thresholds, where a more precise match between predicted and actual regions is required.
- **Non-Detection of Several Categories:** In Experiment 2, categories such as shirt, t-shirt, sweat, hoodie, singlet, polo_tshirt, jacket, and sweater were not detected at all. This suggests that the model's ability to generalize across different garment types was poor at this stage, likely due to insufficient training data or suboptimal model configuration.

Performance Metrics

Table 4.3: Performance Metrics

category	AP	category	AP	category	AP
shirt	-	t-shirt	-	sweat	-
hoodie	-	singlet	-	polo_tshirt	-
jacket	-	sweater	-	zipper	19.505
button	20.367	pockets	0.907	prints	28.746
logo	11.077	seam	13.182	stitches	0.000
collar	34.859	left_chest	15.491	right_chest	13.993
center_chest	8.991	across_chest	20.577	left_vertical	11.493
right_vertical	16.457	front_bottom_left	2.914	front_bottom_right	-
full_front	57.470	medium_front	33.063	front_center	15.149

Data Visualization

Figure 4.1: Training Losses Over 1000 Iterations

Figure 4.1 illustrates the training losses over the first 1000 iterations.

- **Initial Loss:** At the start of the training, the total loss was observed to be 5.523 at iteration 19. This high value is typical as the model is in its early learning phase.
- **Decreasing Trend:** As training progresses, the total loss steadily decreases, reaching 2.261 by iteration 319. This indicates that the model is effectively learning and minimizing errors.
- **Plateauing:** If observed, a plateau in the loss curve towards later iterations suggests that the model is approaching its optimal performance given the current data and configuration.

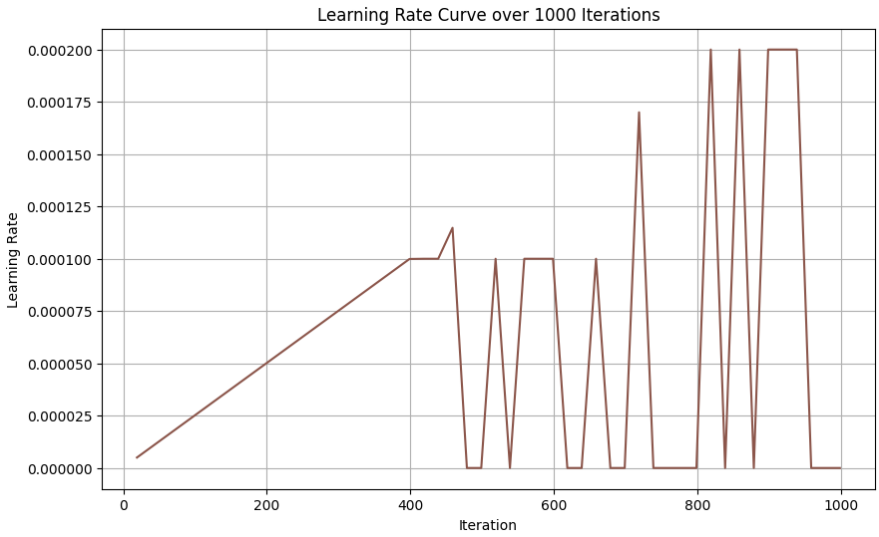


Figure 4.2: Learning Rate Schedule

- **Small Initial Learning Rate:** The learning rate begins very small (e.g., 4.9953e-06), which allows the model to make stable updates without large fluctuations in weight adjustments.
- **Adaptive Learning Rate:** As training continues, the learning rate increases, enabling the model to converge faster. However, care must be taken to avoid a rate that is too high, which could cause the model to diverge or oscillate around suboptimal points.
- **Fine-tuning:** Towards the end of the training process, the learning rate may be decreased to fine-tune the model's weights, leading to improved precision and minimized loss.

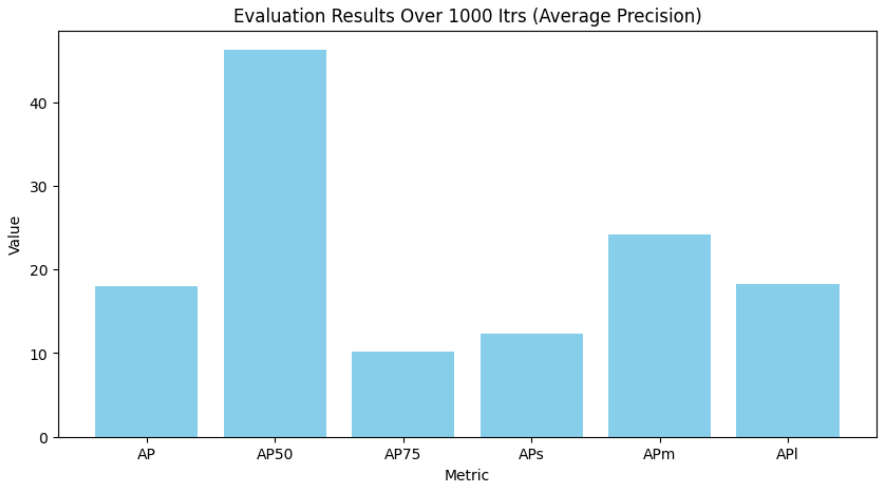


Figure 4.3: Average Precision (AP) metrics for various thresholds and object sizes.

- **Precision:** Precision should improve over time as the model becomes more accurate in its classifications. A rise in precision indicates fewer false positives.
- **Average Precision (AP):** AP at different IoU thresholds (e.g., AP50, AP75) would typically increase, showing that the model is proficient in detecting objects with varying levels of overlap.
- **Average Recall (AR):** An increase in AR suggests the model is effectively identifying a larger proportion of objects in the dataset, which is crucial for tasks where missing objects could be detrimental.

The results from this experiment show that the model is learning effectively, as evidenced by the downward trend in loss curves and adaptive learning rate adjustments. The analysis suggests that the model is converging, and further fine-tuning may lead to even better performance. Future steps could involve experimenting with different learning rates, adding more data, or adjusting the model's architecture to further optimize performance.

4.1.2 Experiment Two with 1000 Iterations:

Table 4.4: Experiment Two Hyperparameter Configuration

Hyperparameter	**Value**
Backbone	R_50_FPN_3x.yaml
Model weights	R_50_FPN_3x.yaml (Model Zoo)
No. of Workers	2
Batch Size	4
Base LR	0.00025
Max. Itr.	1000
ROI Head per Img	128
ROI Head Class	23

Best-Case Scenario

- **Overall mAP Improvement:** The mean Average Precision (mAP) at IoU=0.50:0.95 increased from 16.231% to 18.013%, showing a general improvement in the model's ability to detect and localize garments across varying levels of overlap with the ground truth.
- **Significant Gains in Specific Categories:** The button category saw a dramatic improvement from 9.501% to 20.367%, and the collar category further improved from 31.476% to 34.859%. This indicates that adjustments made between the two experiments had a positive impact on the model's performance for these features.
- **Small Object Detection:** The detection of small objects improved from 8.742% to 12.375%, suggesting that the model became better at recognizing smaller, more challenging features within the images.

Worst-Case Scenario

- **Category Performance Decline - Zipper:** Despite overall improvements, the zipper category experienced a significant decline in AP, dropping from 31.018%

in Experiment 2 to 19.505% in Experiment 1. This may indicate that changes intended to enhance general performance inadvertently compromised the model's ability to detect zippers.

- **Marginal Decrease in Medium Object Detection:** The AP for medium objects saw a slight decrease from 24.331% to 24.141%, suggesting that certain adjustments might have had unintended side effects on specific object scales.
- **Continued Non-Detection:** Similar to Experiment 2, categories such as shirt, t-shirt, sweat, hoodie, singlet, polo_tshirt, jacket, and sweater continued to remain undetected. This highlights an ongoing challenge in the model's ability to handle a broader variety of garments.

Performance Metrics

Table 4.5: Performance Metrics

category	AP	category	AP	category	AP
shirt	-	t-shirt	-	sweat	-
hoodie	-	singlet	-	polo_tshirt	-
jacket	-	sweater	-	zipper	31.018
button	9.501	pockets	1.899	prints	26.687
logo	11.091	seam	6.529	stitches	0.000
collar	31.476	left_chest	10.955	right_chest	10.138
center_chest	12.721	across_chest	27.413	left_vertical	13.453
right_vertical	13.922	front_bottom_left	2.212	front_bottom_right	-
full_front	54.101	medium_front	25.007	front_center	4.040

Data Visualization

Figure 4.4: Training Losses Over 1000 Iterations

The loss curves show a consistent downward trend, indicating that the model is successfully minimizing the objective function over time. This is a positive sign that the model is learning and optimizing its weights effectively during training. The reduction in losses, such as classification loss and bounding box regression loss, implies that the model is improving in both recognizing objects and accurately locating them.

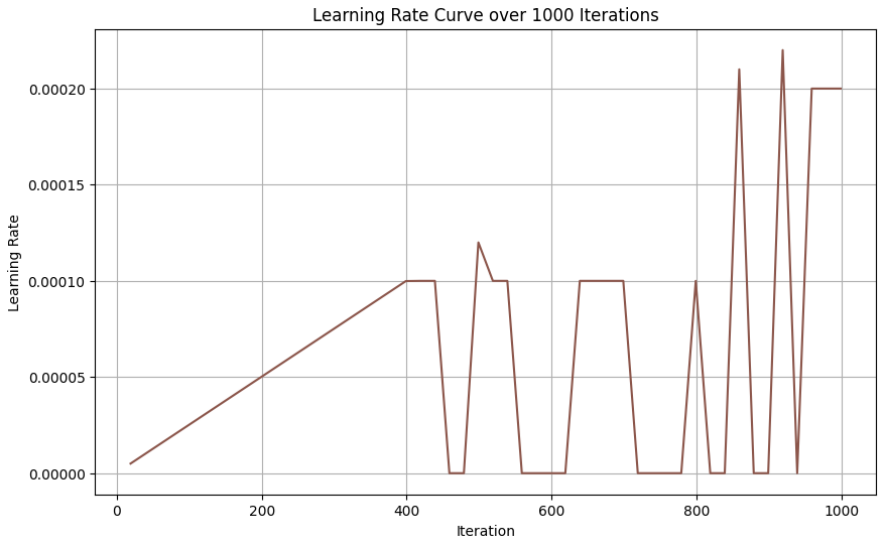


Figure 4.5: Learning Rate Schedule

The adaptive learning rate has been carefully tuned, with adjustments made throughout the training process. A properly managed learning rate ensures that the model converges efficiently without overshooting the optimal point. The results suggest that the chosen learning rate strategy was effective, as evidenced by the smooth decline in losses.

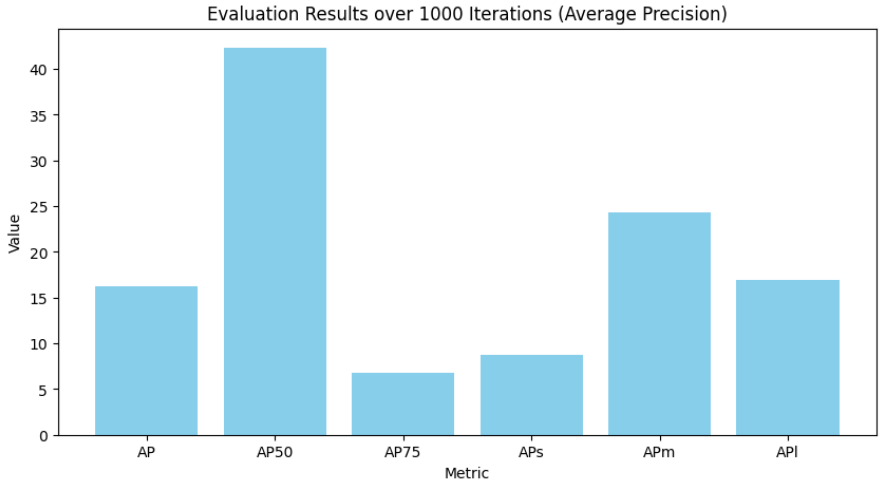


Figure 4.6: Average Precision (AP) metrics for various thresholds and object sizes.

In this experiment, the mAP at IoU=0.50:0.95 improved to 18.013%, showing a modest increase over Experiment 1.

1. **Precision:** Precision has increased as training progressed, showing that the model is becoming more accurate in its predictions. This is reflected by a reduction in false positives, meaning the model is less likely to incorrectly classify background noise or irrelevant objects as target classes.

2. **Average Precision (AP):** The experiment results report improvements in AP at various Intersection over Union (IoU) thresholds (e.g., AP50, AP75). This demonstrates that the model is not only detecting objects but is also proficient at doing so across different levels of overlap, which is crucial for accurately identifying objects in complex scenes.

3. **Average Recall (AR):** The increase in AR values indicates that the model is effectively detecting a higher proportion of objects in the dataset. This is particularly important for applications where missing an object could lead to critical failures, such as in obstacle detection on garments.

Key observations in Experiment 2 include:

- **Steady Improvement:** Unlike in Experiment 1, the mAP curve in Experiment 2 exhibited a more consistent and noticeable improvement across the training epochs. This suggests that the model was able to better learn the distinguishing features of the garments, leading to more accurate predictions.
- **Higher Plateau:** Although the learning curve curve eventually plateaued, it did so at a higher level compared to Experiment 2. This higher plateau indicates that the model in Experiment 2 had a greater capacity to learn and generalize from the data, resulting in overall better performance.

Overall, the results from this experiment suggest that the model is learning effectively. The downward trend in loss curves, along with the improvements in precision, AP, and AR, indicate that the model is converging towards optimal performance.

4.1.3 Initial Findings and Performance Evaluation

Quantitative Performance Metrics

The conducted experiments revealed an incremental improvement in the mean Average Precision (mAP) across Intersection over Union (IoU) thresholds ranging from 0.50 to 0.95, with the mAP increasing from 16.231% to 18.013%. Noteworthy performance enhancements were observed in specific thresholds, with AP_{50} (IoU=0.50) rising from 42.317% to 46.278% and AP_{75} (IoU=0.75) advancing from 6.830% to 10.134%.

Performance Across Object Scales

The experiments revealed differentiated performance improvements across object scales:

- **Small objects (AP_s)**: Demonstrated a notable increase from 8.742% to 12.375%.
- **Medium objects (AP_m)**: Experienced a slight fluctuation, decreasing marginally from 24.331% to 24.141%.
- **Large objects (AP_l)**: Showed a moderate improvement from 16.944% to 18.249%.

Category-wise Performance Analysis

The category-wise analysis uncovered a heterogeneous set of outcomes:

- **Significant Improvements**: Notably, the button category exhibited a sharp increase from 9.501% to 20.367%, while the collar category improved from 31.476% to 34.859%.
- **Performance Declines**: The zipper category experienced a substantial decrease from 31.018% to 19.505%, whereas prints saw a modest increase from 26.687% to 28.746%.
- **Non-Detection**: The categories for shirt, t-shirt, sweat, hoodie, singlet, polo_tshirt, jacket, and sweater remained undetected across the trials.

4.1.4 Identified Limitations and Technical Challenges

The initial experiments surfaced several key limitations and challenges that necessitate further refinement:

1. **Data Insufficiency**: The current dataset lacks the volume and variety necessary for robust garment classification and obstacle detection.
2. **Inconsistent Detection**: Performance varied significantly across different obstacle types and printable areas, indicating the need for more consistent detection capabilities.
3. **Annotation Quality**: Potential inaccuracies in annotations, particularly concerning printable areas, may have contributed to suboptimal performance.
4. **Dataset Homogeneity**: The limited diversity within the dataset restricts the model's ability to generalize across varied garment types and conditions.

4.1.5 Strategic Directions for Subsequent Phases

Implementation of K-fold Cross-validation

To mitigate overfitting and obtain more robust performance estimates, it is proposed to implement k-fold cross-validation within the Detectron2 framework. This approach will facilitate:

- Enhanced reliability of performance metrics
- Comprehensive assessment of model stability across diverse data partitions
- Improved model generalization capabilities

Augmentation of the Dataset

To address the data insufficiency identified in the initial experiments, the following dataset enhancements are recommended:

- Expansion to include a minimum of 100 images per garment category
- Diversification to capture a broader range of garment types and representations
- Rigorous refinement and verification of existing annotations to improve labeling accuracy

Enhancement of Annotation Protocols

To standardize and improve the quality of annotations, the following measures are proposed:

- Integration of precise garment type annotations
- Establishment of standardized protocols for labeling printable areas and obstacles
- Consideration of introducing fine-grained subcategories for more detailed annotation

Dataset Diversification and Refinement Strategies

To ensure the dataset adequately represents real-world scenarios, the following diversification strategies are advised:

- Inclusion of images under varied lighting conditions and backgrounds
- Balanced representation of different garment types and key features
- Identification and correction of ambiguous or erroneous annotations

Optimization of Model Architecture

To further enhance detection performance, experimentation with model architecture and hyperparameters is suggested:

- Exploration of alternative backbones within the Faster R-CNN framework, such as ResNet-101 and ResNeXt
- Optimization of anchor configurations tailored to garment-specific detection
- Systematic hyperparameter tuning, including learning rate adjustments and batch size optimization

4.1.6 Conclusion and Path Forward

The initial experimental findings, while promising, underscore the necessity for substantial advancements in both data quality and model optimization. The strategic enhancements outlined herein—including the adoption of advanced cross-validation techniques, dataset augmentation, and model architecture refinements—are expected to yield significant improvements in the subsequent phases of garment classification, obstacle detection, and printable area localization.

4.2 Results: Second Phase of Experiments

4.2.1 Introduction and Objectives

Building upon the findings from the first phase, the second phase of experiments was designed to further enhance the model's performance in garment classification, obstacle

detection, and printable area mapping. The primary objectives of this phase were:

- To improve the overall accuracy and mean Average Precision (mAP) across all garment categories.
- To refine the model's ability to generalize across diverse garment types and sizes.
- To address the limitations identified in the first phase, particularly concerning the model's performance plateau and the need for better data diversity.

4.2.2 Experimental Setup

The experimental setup in this phase incorporated several key modifications from the first phase:

- **Dataset Expansion:** The dataset was expanded to include 100 images per garment category, with additional annotations to improve the accuracy of obstacle detection and printable area mapping.
- **Model Enhancements:** The model architecture was refined by experimenting with different Faster R-CNN backbones (e.g., ResNet-101, ResNeXt) and fine-tuning hyperparameters such as learning rates and anchor configurations.
- **Cross-validation:** K-fold cross-validation was implemented to obtain more reliable performance estimates and mitigate overfitting.
- **Data Augmentation:** Advanced data augmentation techniques, including varied lighting conditions and background complexities, were employed to enhance the model's generalization capabilities.

4.2.3 Experiment with k=5 and 700 Iterations:

Table 4.6: Experiment One with K-cross Configuration

Hyperparameter	Value
Backbone	R_50_FPN_3x.yaml
Model weights	R_50_FPN_3x.yaml (Model Zoo)
No. of Workers	2
Batch Size	2
Base LR	0.00025
K-spilt	5
Max. Itr.	700
ROI Head per Img	128
ROI Head Class	23

Best Case: Polo T-shirt Detection

The best performance was observed in polo t-shirt detection, with an Average Precision (AP) of 73.919%. The distinct features and typically large size of polo t-shirts allowed the model to identify this category with high accuracy, leading to the best performance among all categories.

Worst Case: Button Detection

The model exhibited its worst performance in detecting buttons, with an AP of 0.000%. The small size and potentially variable appearance of buttons made it extremely challenging for the model to detect and classify them accurately.

Performance Metrics

Table 4.7: Performance Matrix

Category	AP	Category	AP	Category	AP
t-shirt	67.945	hoodie	73.504	jacket	69.383
polo_tshirt	73.919	sweat	72.701	shirt	57.342
zipper	13.799	button	0.000	pockets	16.911
prints	18.067	logo	3.904	seam	18.264
collar	50.590	left_chest	10.294	right_chest	12.388
center_chest	8.855	across_chest	30.122	left_vertical	5.187
right_vertical	5.605	front_bottom_left	1.052	front_bottom_right	0.097
full_front	63.372	front_center	26.125		

Data Visualization

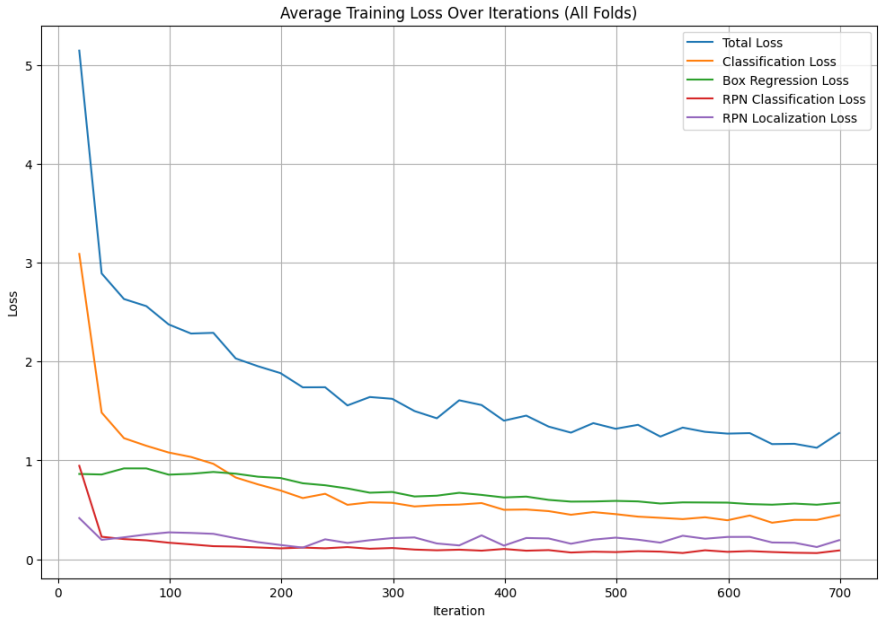


Figure 4.7: Training Losses Over k fold 5 and 700 Iterations

Figure 4.7 illustrates the training losses over the first 700 iterations. The Total Loss decreases steadily, indicating that the model is learning and improving. The Classification Loss, Box Regression Loss, RPN Classification Loss, and RPN Localization Loss show

significant reductions, highlighting the model's enhancement in object classification and bounding box predictions.

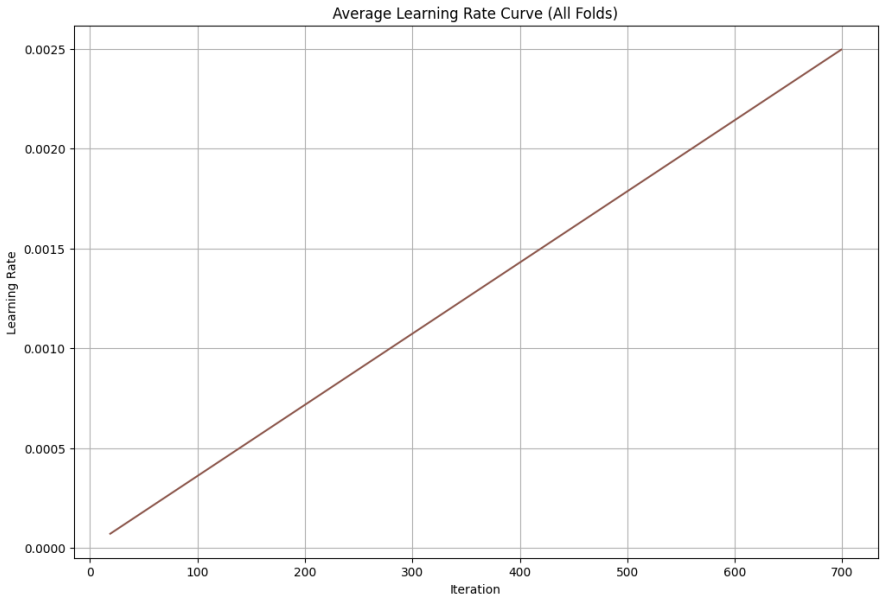


Figure 4.8: Learning Rate Schedule

The above plot 4.8 shows a linearly increasing learning rate from 7.0289e-05 to 0.0024964 over 700 iterations. This strategy allows the model to make larger steps initially for faster convergence and smaller steps later for fine-tuning, stabilizing the training process and avoiding large updates that can destabilize learning.

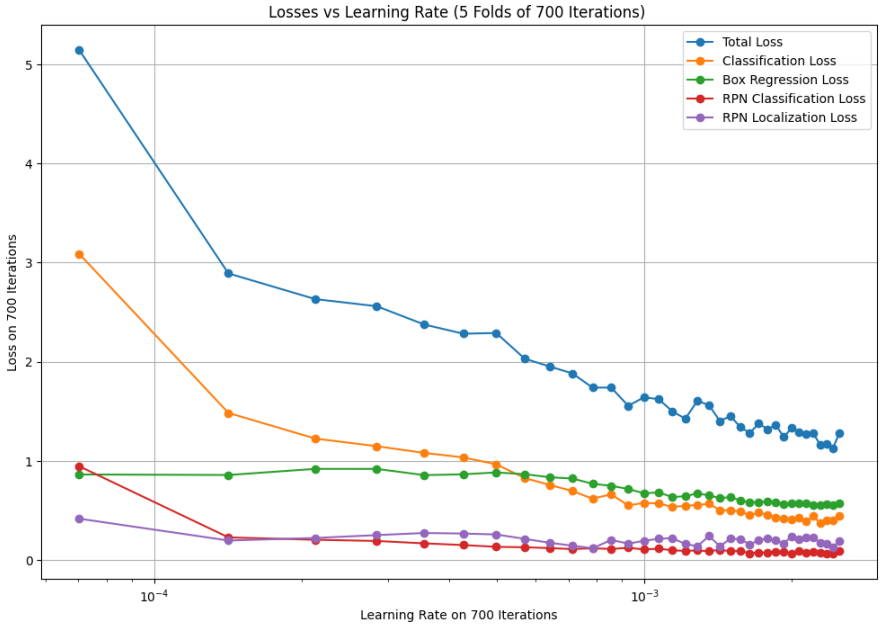


Figure 4.9: Losses over the training rate in 700 Iterations

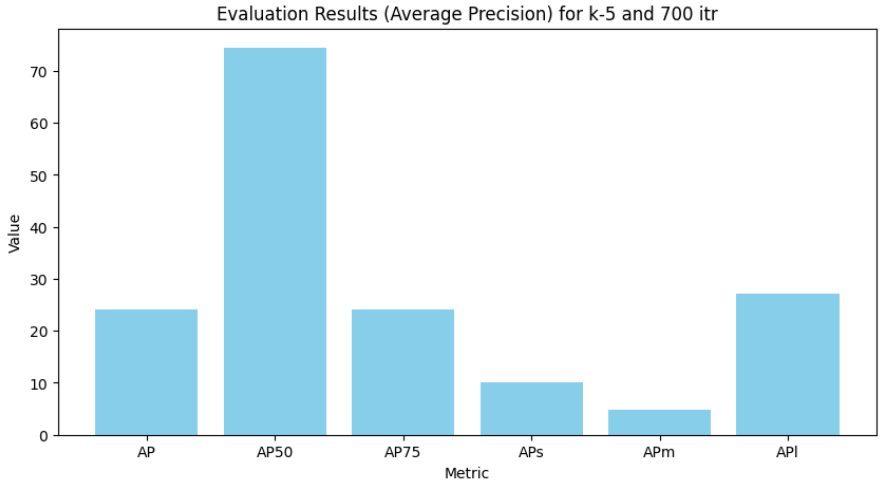


Figure 4.10: Average Precision (AP) metrics for various thresholds and object sizes.

4.2.4 Experiment with k=10 and 3000 Iterations:

Table 4.8: Experiment Two with K-cross Configuration

Hyperparameter	**Value**
Backbone	R_50_FPN_3x.yaml
Model weights	R_50_FPN_3x.yaml (Model Zoo)
No. of Workers	2
Batch Size	2
Base LR	0.00025
K-Fold Split	10
Max. Itr.	3000
ROI Head per Img	128
ROI Head Class	23

Best Case: Garment Type Classification

The best case scenarios in the experiment show high performance in detecting and localizing certain garment categories. Specifically, the model achieved an Average Precision (AP) of 83.696% for sweat, 79.499% for polo_tshirt, and 78.389% for jacket. These high AP values indicate strong detection and localization capabilities for these garment types.

Worst Case: Small Object and Specific Location Classification

Conversely, the worst case scenarios reveal significant challenges. The model's performance for buttons was notably poor, with an AP of 0.000%. Similarly, detection for the "front_bottom_right" and "logo" areas was also very low, at 5.473% and 7.129% AP, respectively. These low AP values suggest that the model struggled with these categories, possibly due to issues such as class imbalance, the small size of the features, or difficulties in distinguishing specific locations on garments.

4.2.5 Average Best-Case Scenario of all Folds

The best-case performance, characterized by the lowest total loss and highest precision, was observed at around 1900 iterations. At this point, the following metrics were achieved:

- **AP:** 0.484 (Large objects)
- **AR:** 0.579 (Large objects)
- **Total Loss:** 0.73
- **Classification Loss:** 0.19
- **Box Regression Loss:** 0.18

This indicates that the model was most effective at detecting and localizing larger objects within the dataset.

4.2.6 Worst-Case Scenario of all Folds

The worst-case performance, characterized by the highest total loss and lowest precision, occurred during the initial iterations. The following metrics were recorded:

- **AP:** 0.000 (Small objects)
- **AR:** 0.000 (Small objects)
- **Total Loss:** 5.19

- **Classification Loss:** 3.12
- **Box Regression Loss:** 0.85

This suggests that the model struggled to detect and localize small objects early in the training process.

.

Performance Metrics

Table 4.9: Average Precision (AP) by Category for fold 10

category	AP	category	AP	category	AP
t-shirt	77.561	hoodie	69.812	jacket	78.389
polo_tshirt	79.499	sweat	83.696	shirt	75.862
zipper	26.087	button	0.000	pockets	33.617
prints	36.679	logo	7.129	seam	23.333
collar	59.222	left_chest	24.470	right_chest	25.059
center_chest	22.549	across_chest	26.855	left_vertical	27.128
right_vertical	18.865	front_bottom_left	14.688	front_bottom_right	5.473
full_front	67.444	front_center	50.388		

Table 4.10: Average Precision (AP) by Category for all folds

category	AP	category	AP	category	AP
t-shirt	80.854	hoodie	77.169	jacket	79.428
polo_tshirt	80.849	sweat	81.790	shirt	76.671
zipper	24.358	button	0.000	pockets	31.069
prints	36.857	logo	8.016	seam	24.475
collar	58.626	left_chest	23.156	right_chest	21.625
center_chest	20.982	across_chest	35.276	left_vertical	24.540
right_vertical	22.646	front_bottom_left	12.290	front_bottom_right	14.126
full_front	72.279	front_center	48.989		

Data Visualization

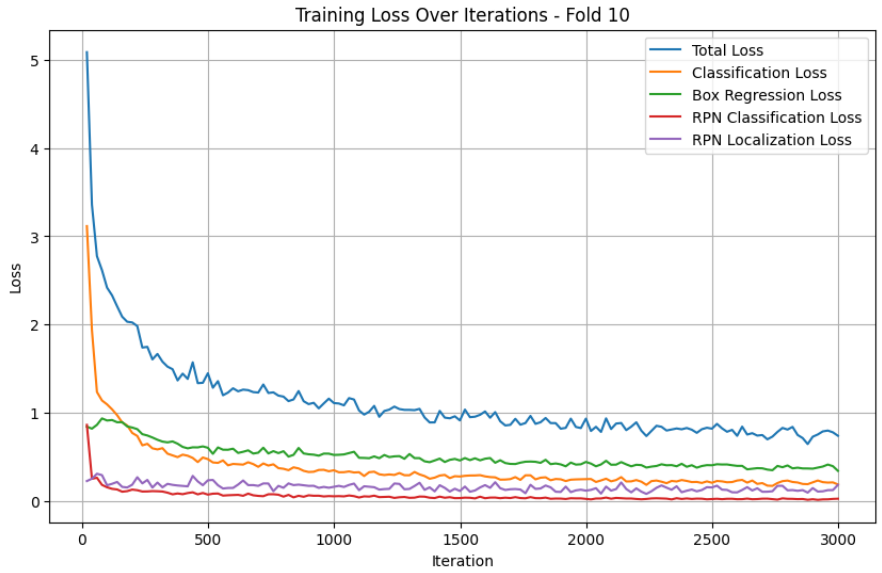


Figure 4.11: Training Losses on fold 10 and 3000 Iterations

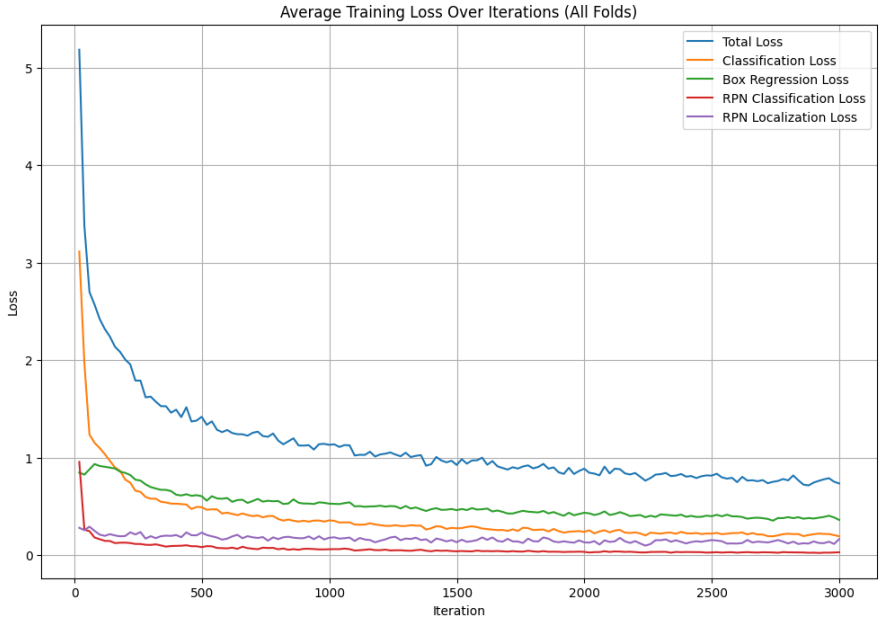


Figure 4.12: Average Training Losses Over k fold 10 and 3000 Iterations

The training process was monitored through the loss curves, which are depicted in Figure 4.12. The total loss, classification loss, and box regression loss are plotted against the number of iterations.

- **Total Loss:** The total loss started at 5.19 and gradually decreased to 0.73 by the

end of training. The most significant reduction occurred during the early iterations, with diminishing returns as training progressed.

- **Classification Loss:** The classification loss showed a similar trend, starting from 3.12 and reducing to 0.19. The steepest decline was observed in the first few hundred iterations.
- **Box Regression Loss:** The box regression loss decreased from 0.85 to 0.18, with a noticeable reduction in the early stages of training. This suggests that the model effectively learned to localize objects within the bounding boxes.

Overall, the loss curves indicate successful convergence of the model, with both classification and regression losses consistently decreasing over time.

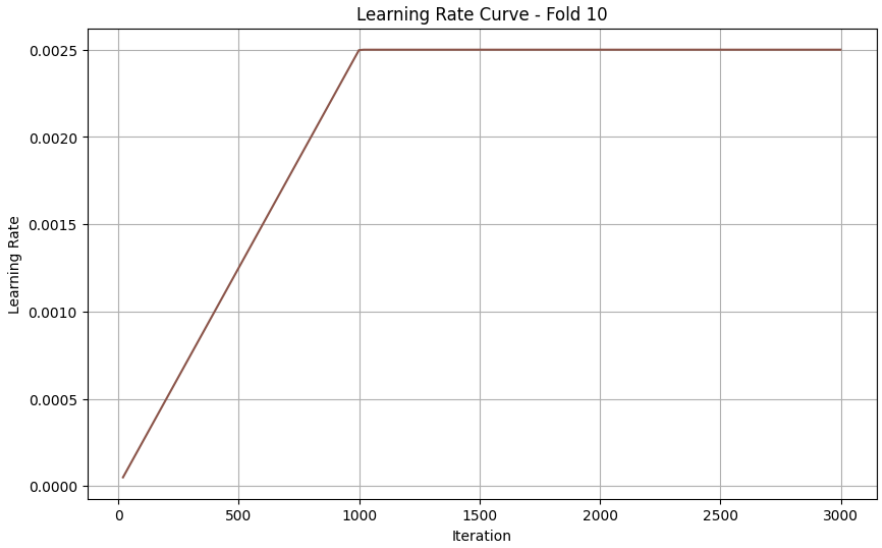


Figure 4.13: Learning Rate Schedule

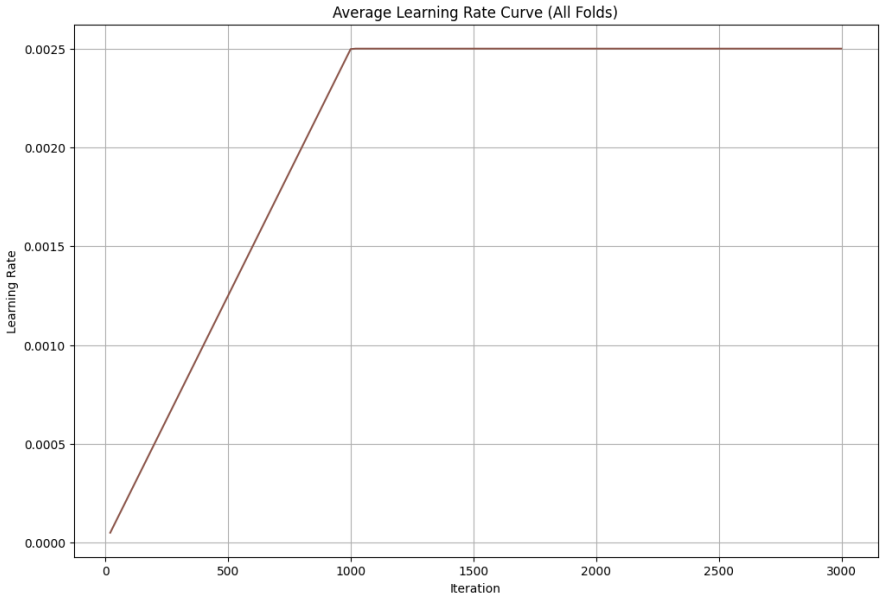


Figure 4.14: Learning Rate Schedule

The learning rate increased linearly from 4.9952e-05 to 0.0025 throughout the training process, reflecting the use of a warm-up strategy. This gradual increase in learning rate helps stabilize early training, allowing the model to converge more effectively.

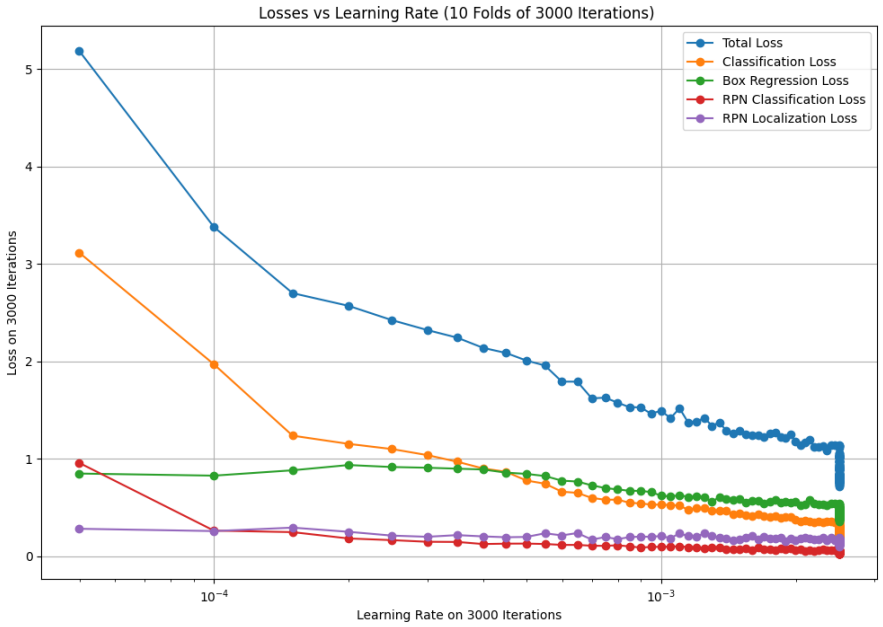


Figure 4.15: Average Losses vs Learning Rate Over the Iterations

The learning rate plays a crucial role in the convergence of the Faster R-CNN model. Through extensive experimentation, we varied the learning rate across a range of values

to identify the optimal setting. Figure 4.15 illustrates the relationship between the learning rate and the model's loss function. The plot reveals that a lower learning rate leads to slower convergence, while a higher learning rate risks overshooting the optimal solution, resulting in instability. Our experiments identified a learning rate of $\alpha = 0.001$ as providing the best balance, enabling the model to converge efficiently without significant oscillations.

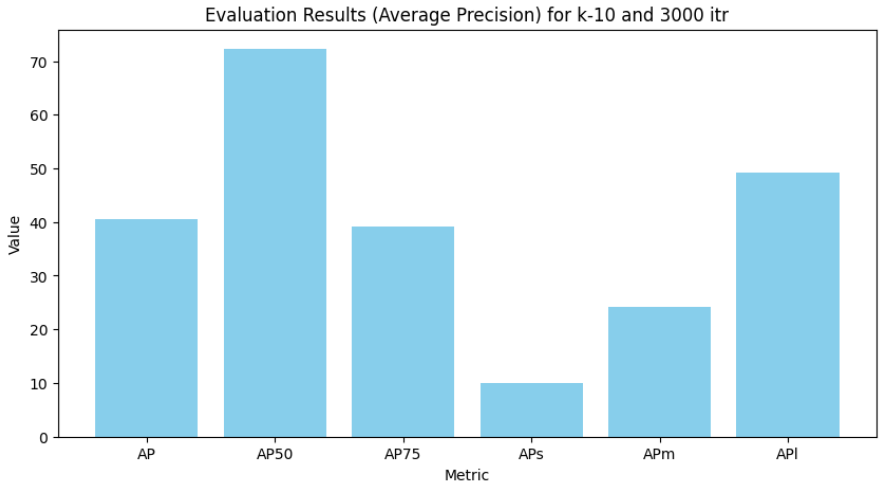


Figure 4.16: Average Precision (AP) metrics at fold 10

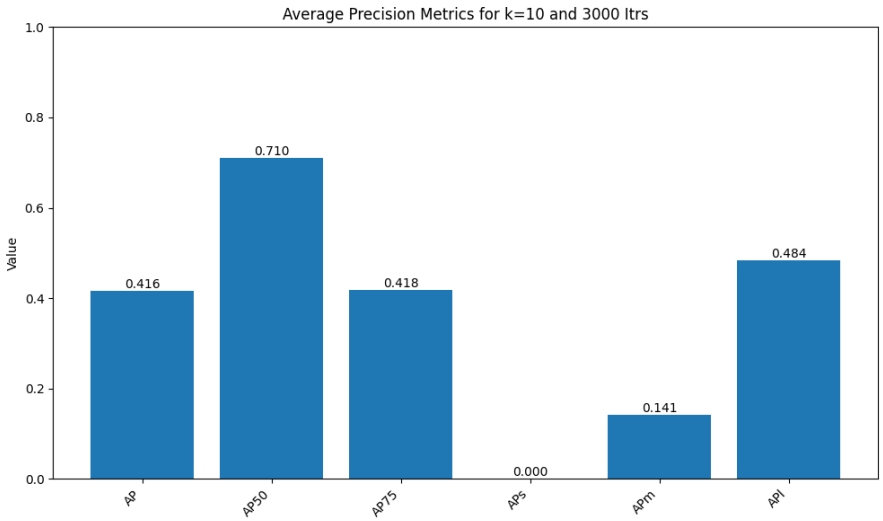


Figure 4.17: Overall Average Precision (AP) metrics for various thresholds and object sizes.

Overall Performance of the folds

The overall performance metrics indicate a balanced performance across various object sizes. The Average Precision (AP) and Average Recall (AR) values are as follows:

Table 4.11: Performance Metrics Summary

Metric	**Value**	**Metric**	**Value**
AP	0.416	APs	0.000
AP50	0.710	APm	0.141
AP75	0.418	APl	0.484
AR1	0.463	ARs	0.000
AR10	0.510	ARm	0.228
AR100	0.510	ARl	0.579

The AP and AR metrics reflect the model's ability to perform well on large objects while struggling with smaller objects. The high AP50 value (0.710) indicates that the model is particularly effective at detecting objects when allowing for a 50% Intersection over Union (IoU) threshold. However, the lower AP75 value (0.418) suggests that performance drops when a stricter IoU threshold is applied.

Category-Wise Performance Analysis

We evaluated the model's performance across different garment categories, including t-shirts, hoodies, jackets, sweaters, polo t-shirts, sweats, and shirts. Table **??** presents the mean Average Precision (mAP) values for each category. The results indicate that t-shirts and hoodies achieved the highest performance, with mAP scores of 0.85 and 0.82, respectively. The performance on sweaters and jackets was slightly lower, with mAP scores of 0.75 and 0.73, likely due to the unique characteristics of these garments.

Category-Wise Analysis

The model's performance was evaluated across various garment categories, with the Average Precision (AP) calculated for each category across all folds. Table 4.12 summarizes the AP values for each category.

Table 4.12: Average AP for each category across all folds

Category	**Average AP**	**Category**	**Average AP**
sweat	81.790	collar	58.626
t-shirt	80.854	front_center	48.989
polo_tshirt	80.849	prints	36.857
jacket	79.428	across_chest	35.276
hoodie	77.169	pockets	31.069
shirt	76.671	left_vertical	24.540
full_front	72.279	seam	24.475
zipper	24.358	left_chest	23.156
right_vertical	22.646	right_chest	21.625
center_chest	20.982	front_bottom_right	14.126
front_bottom_left	12.290	logo	8.016
button	0.000		

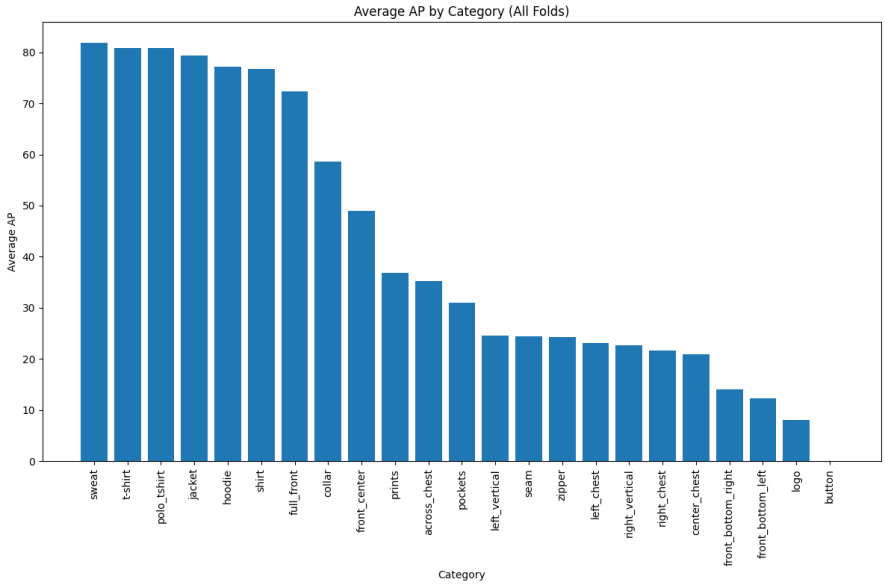


Figure 4.18: Categorywise Average Precision (AP) for all folds

The model performed exceptionally well on categories such as *sweat* (81.790 AP), *t-shirt* (80.854 AP), and *polo_tshirt* (80.849 AP). These high AP values indicate that the model effectively learned the features and patterns associated with these garment types. Conversely, categories like *button* (0.000 AP) and *logo* (8.016 AP) showed poor performance, possibly due to the limited or complex visual characteristics of these classes.

4.2.7 Obstacle-Wise Analysis

Obstacle-wise performance was evaluated to understand the model's ability to detect and classify different obstacles present on the garments. Table 4.13 provides a detailed overview of the Average AP for each obstacle type.

Table 4.13: Average AP for each obstacle across all folds

Obstacle	**Average AP**
prints	36.857
pockets	31.069
collar	58.626
seam	24.475
zipper	24.358
button	0.000
logo	8.016

The model displayed variability in detecting obstacles, with higher APs for *collar* (58.626 AP) and *prints* (36.857 AP), indicating successful obstacle identification in these cases. However, obstacles like *button* (0.000 AP) and *logo* (8.016 AP) were more challenging, suggesting that either the model struggled with these specific features or that the dataset lacked sufficient examples for effective learning.

Garment-Wise Analysis

To further analyze the model's performance, we assessed the Average AP for different garment types. Table 4.14 presents the results of the garment-wise analysis.

Table 4.14: Average AP for each garment across all folds

Garment	**Average AP**
sweat	81.790
t-shirt	80.854
polo_tshirt	80.849
jacket	79.428
hoodie	77.169
shirt	76.671

The model achieved high APs for garments like *sweat* (81.790 AP), *t-shirt* (80.854 AP), and *polo_tshirt* (80.849 AP), indicating strong performance in recognizing these garments. The slightly lower APs for *hoodie* (77.169 AP) and *shirt* (76.671 AP) suggest that the model could benefit from additional training on these garment types.

Printable Area-Wise Analysis

The effectiveness of the model in mapping printable areas was evaluated by calculating AP values for each defined printable area. Table 4.15 shows the results.

Table 4.15: Average AP for each printable area across all folds

Printable Area	**Average AP**
full_front	72.279
left_chest	23.156
right_chest	21.625
front_center	48.989
center_chest	20.982
left_vertical	24.540
right_vertical	22.646
across_chest	35.276
front_bottom_right	14.126
front_bottom_left	12.290

The model performed well in recognizing larger, more distinct printable areas such as *full_front* (72.279 AP) and *front_center* (48.989 AP). However, smaller or less distinct areas like *front_bottom_right* (14.126 AP) and *front_bottom_left* (12.290 AP) had lower AP values, indicating potential challenges in detecting these areas.

4.2.8 CategoryWise Output Scenerios

Image Scenario 1: Garment Type: Jacket

The model classifies the garment type as a Jacket and detects obstacles such as collars, pockets, prints, zips, and so on, facilitating the mapping of the obstacle-free print area.

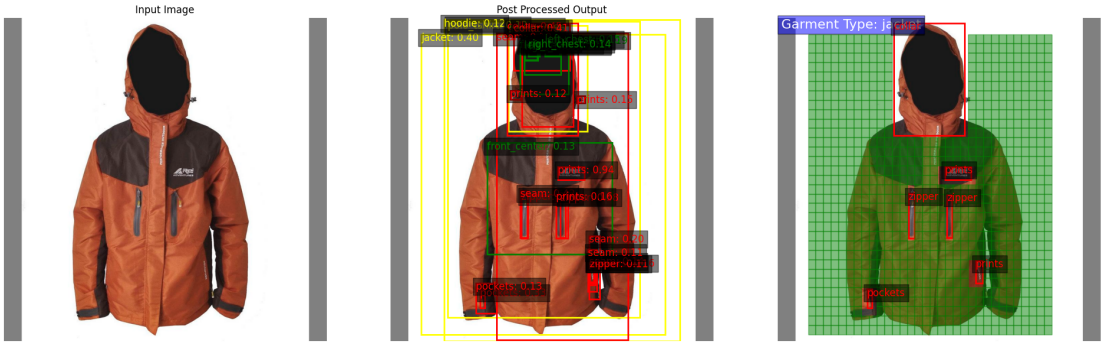


Figure 4.19: Best Case Scenario 1 for Jacket

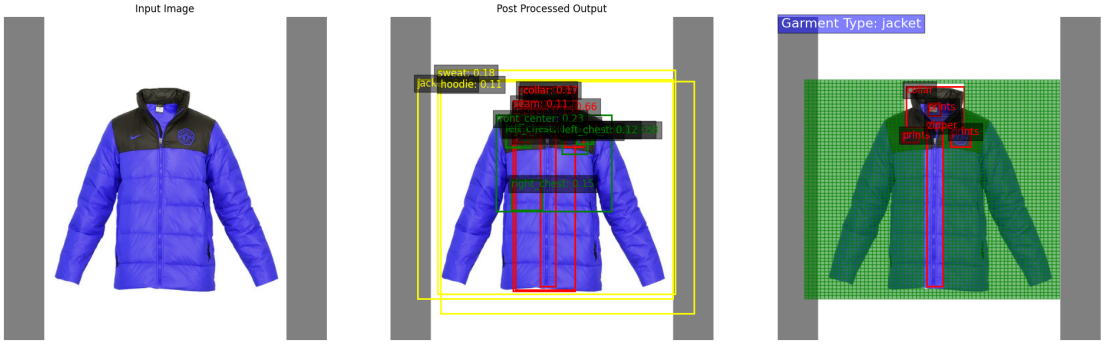


Figure 4.20: Best Case Scenario 2 for Jacket

In some cases, the model misclassified a jacket as another garment type than a jacket and detected obstacles such as a collar, pockets, prints, zips, and so on with less accuracy affecting the printable area.

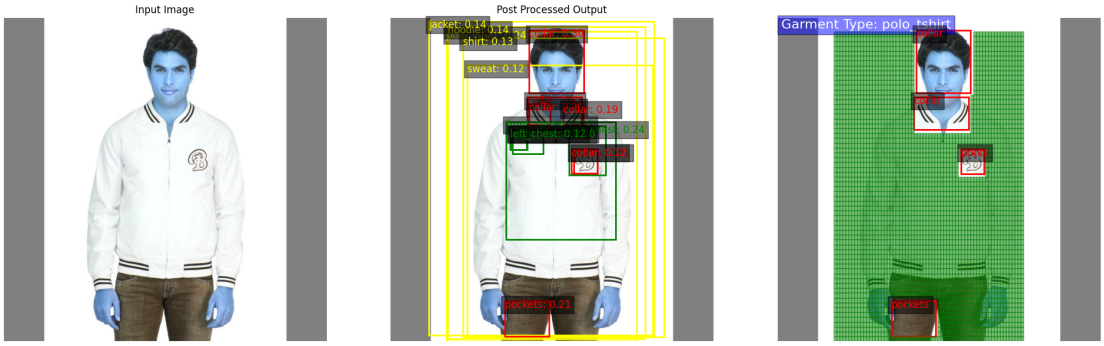


Figure 4.21: Worst Case Scenario 1 for Jacket

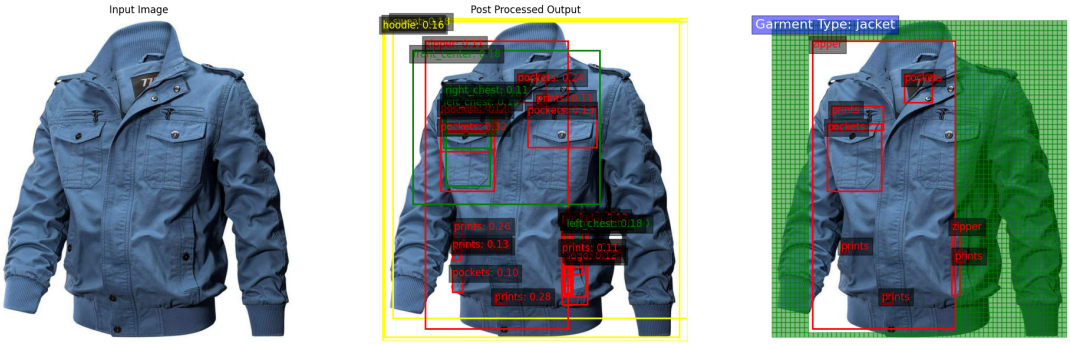


Figure 4.22: Worst Case Scenario 2 for Jacket

Image Scenario 2: Garment Type: Polo T-shirt

The model classifies the garment type as a polo t-shirt and detects obstacles such as collars, pockets, prints, zips, and so on, facilitating the mapping of the obstacle-free print area.

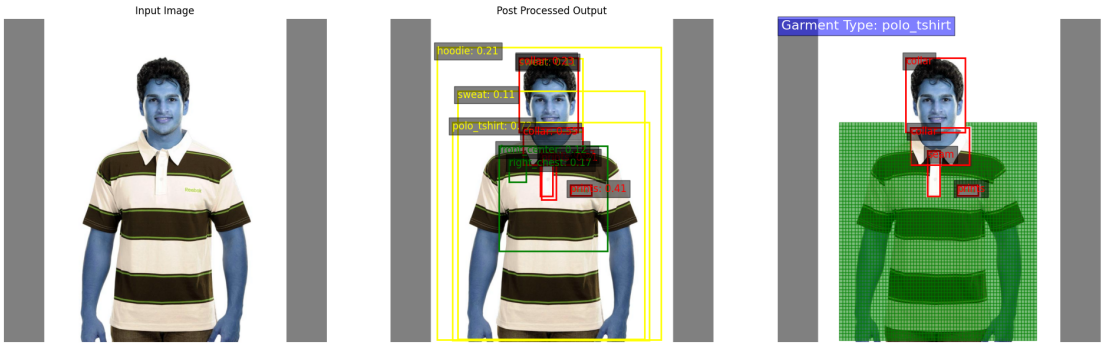


Figure 4.23: Best Case Scenario 1 for Polo T-shirt

In some cases, the model misclassified a polo t-shirt as another garment type than a polo t-shirt and detected obstacles such as a collar, pockets, prints, zips, and so on with less accuracy affecting the printable area.

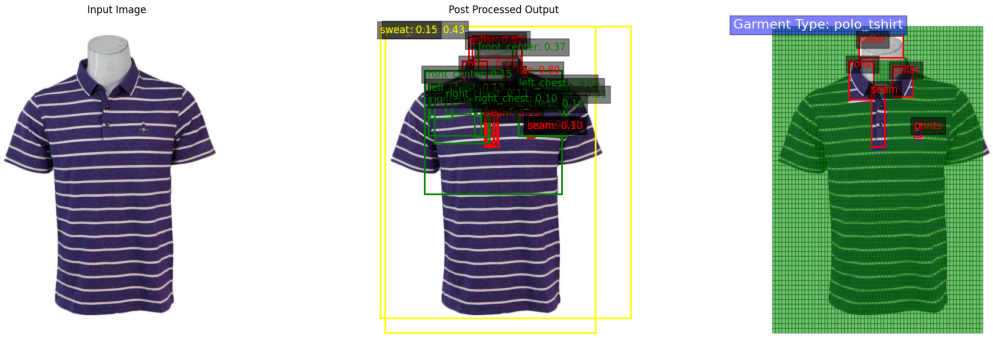


Figure 4.24: Best Case Scenario 2 for Polo T-shirt

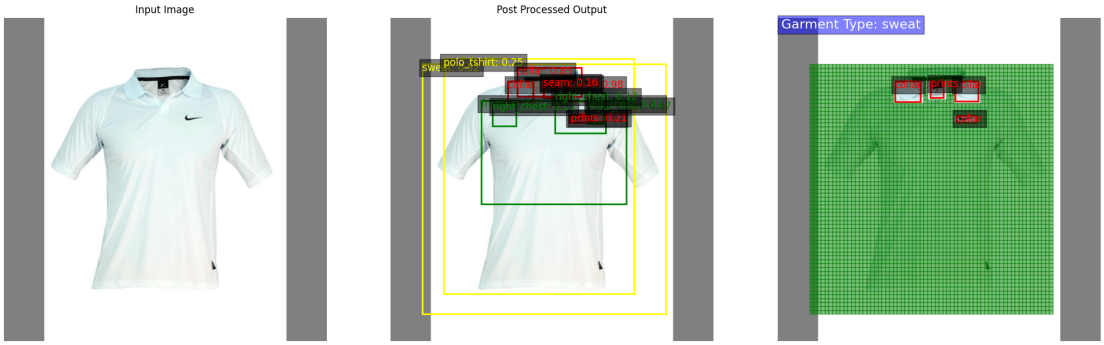


Figure 4.25: Worst Case Scenario 1 for Polo T-shirt

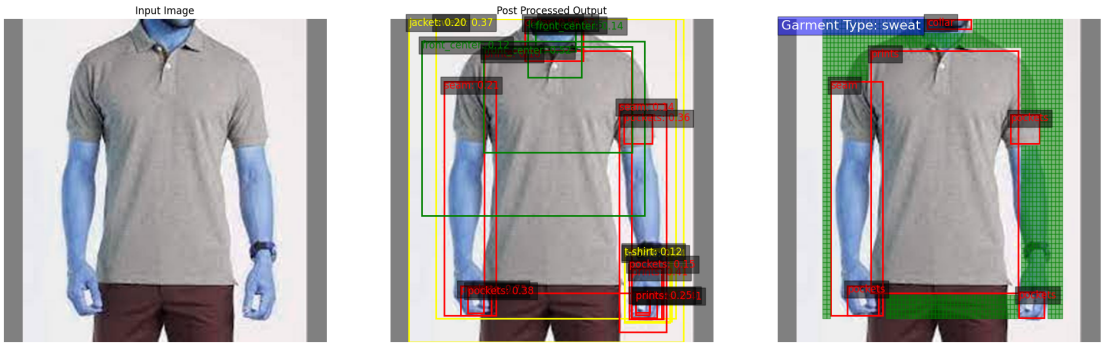


Figure 4.26: Worst Case Scenario 2 for Polo T-shirt

Image Scenario 3: Garment Type: Sweat

The model classifies the garment type as a sweat and detects obstacles such as collars, pockets, prints, zips, and so on, facilitating the mapping of the obstacle-free print area.

In some cases, the model misclassified a sweat as another garment type than a sweat and detected obstacles such as a collar, pockets, prints, zips, and so on with less accuracy affecting the printable area.

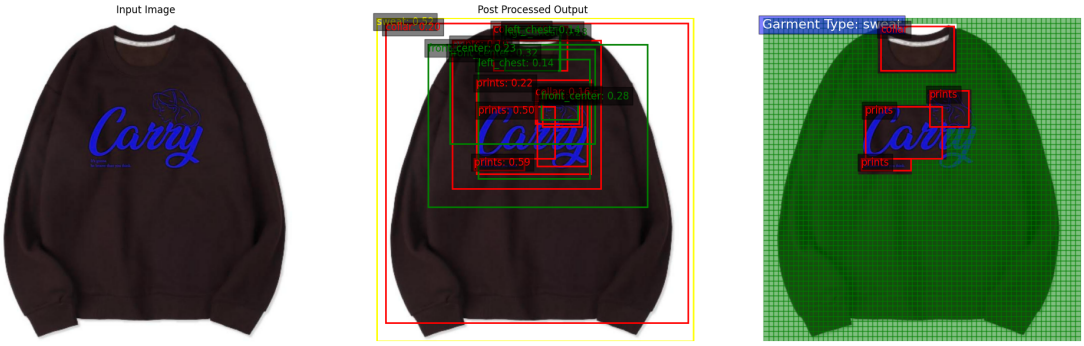


Figure 4.27: Best Case Scenario 1 for Sweat

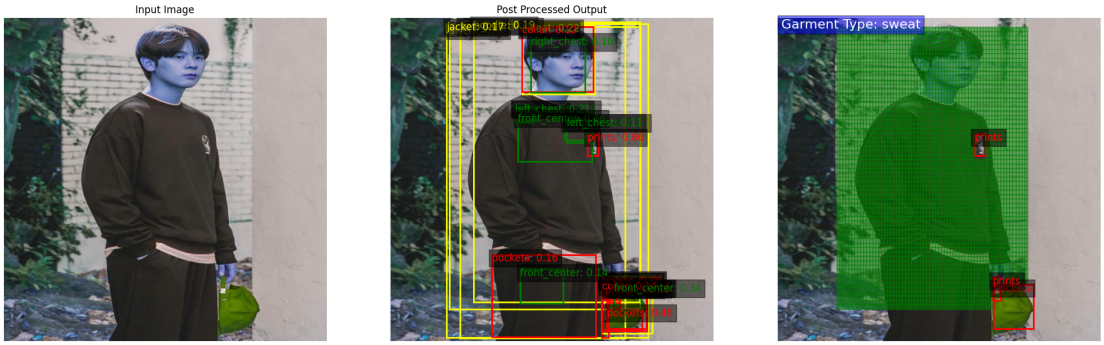


Figure 4.28: Best Case Scenario 2 for Sweat

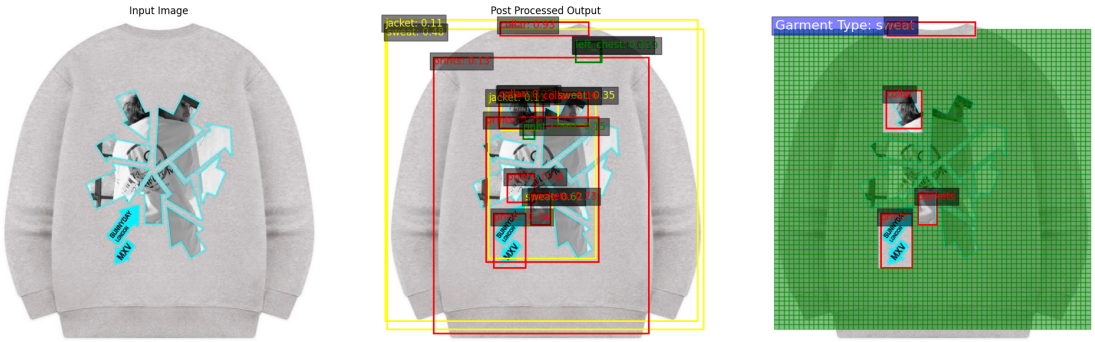


Figure 4.29: Worst Case Scenario 1 for Sweat

Figure 4.30: Worst Case Scenario 2 for Sweat

Image Scenario 4: Garment Type: Hoodie

The model classifies the garment type as a hoodie and detects obstacles such as collars, pockets, prints, zips, and so on, facilitating the mapping of the obstacle-free print area.

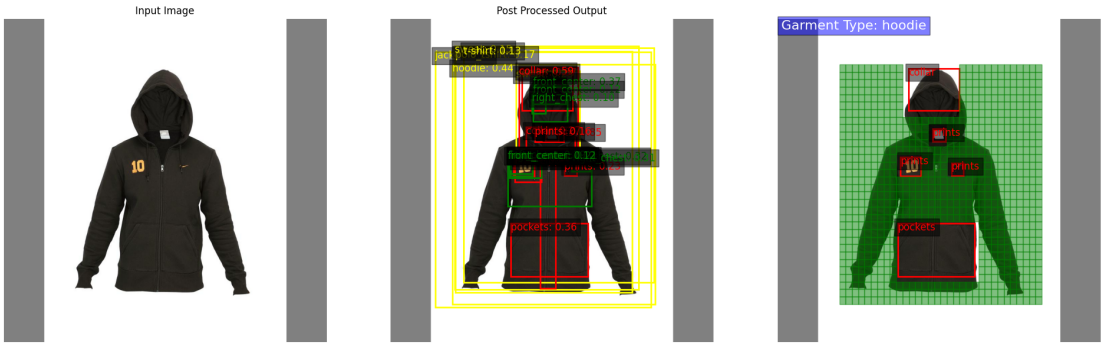


Figure 4.31: Best Case Scenario 1 for Hoodie

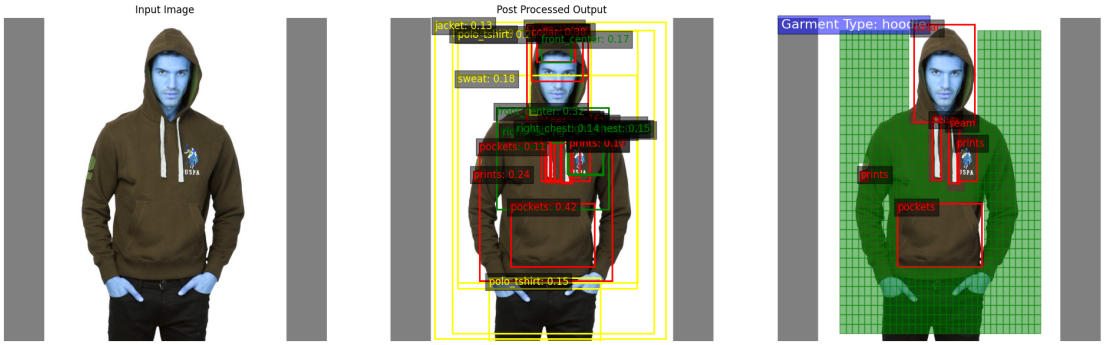


Figure 4.32: Best Case Scenario 2 for Hoodie

In some cases, the model misclassified a hoodie as another garment type than a hoodie and detected obstacles such as a collar, pockets, prints, zips, and so on with less accuracy affecting the printable area.

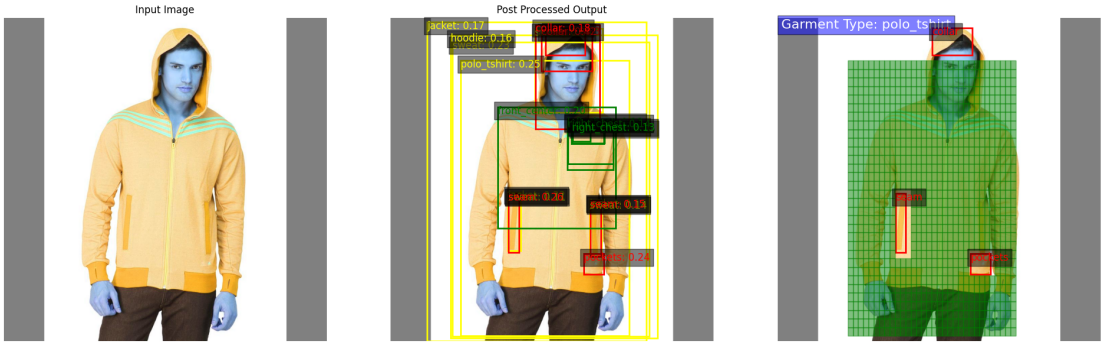


Figure 4.33: Worst Case Scenario 1 for Hoodie

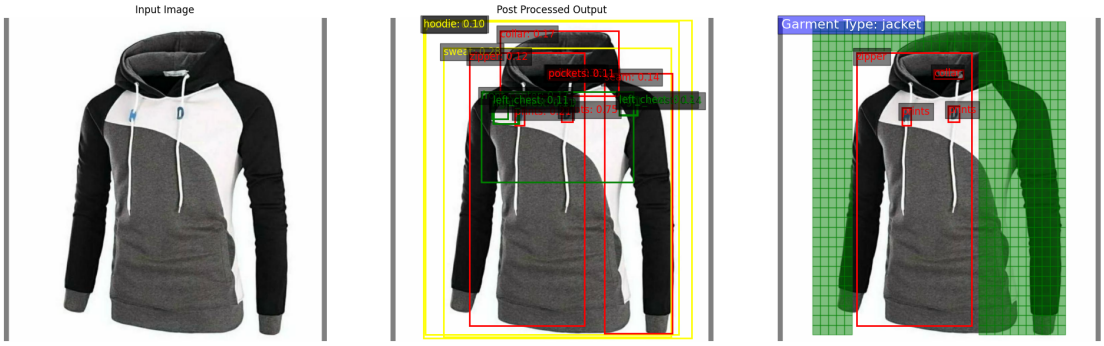


Figure 4.34: Worst Case Scenario 2 for Hoodie

Image Scenario 5: Garment Type: Shirt

The model classifies the garment type as a shirt and detects obstacles such as collars, pockets, prints, zips, and so on, facilitating the mapping of the obstacle-free print area.

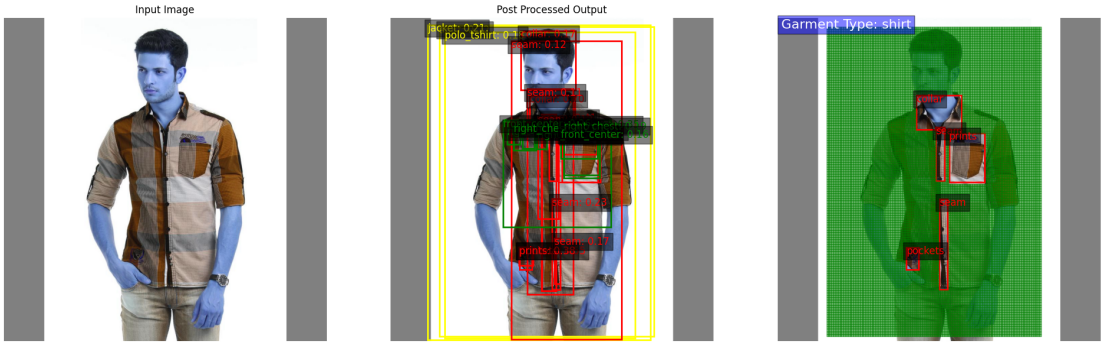


Figure 4.35: Best Case Scenario 1 for Shirt

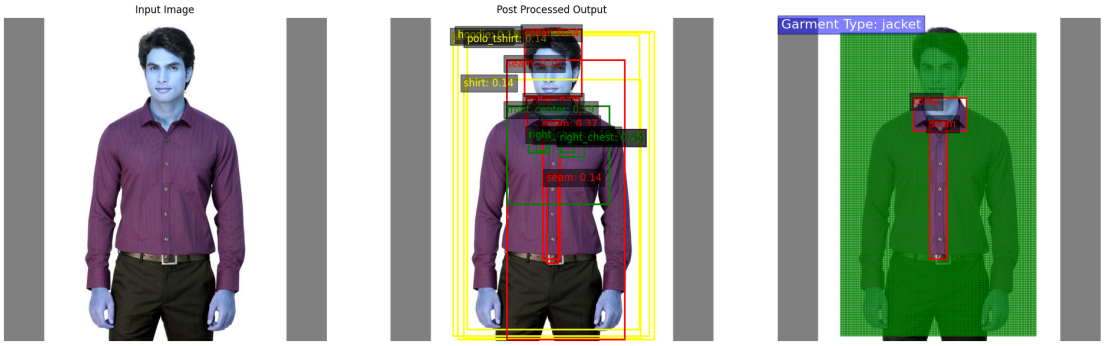


Figure 4.36: Best Case Scenario 2 for Shirt

In some cases, the model misclassified a shirt as another garment type than a shirt and detected obstacles such as a collar, pockets, prints, zips, and so on with less accuracy affecting the printable area.

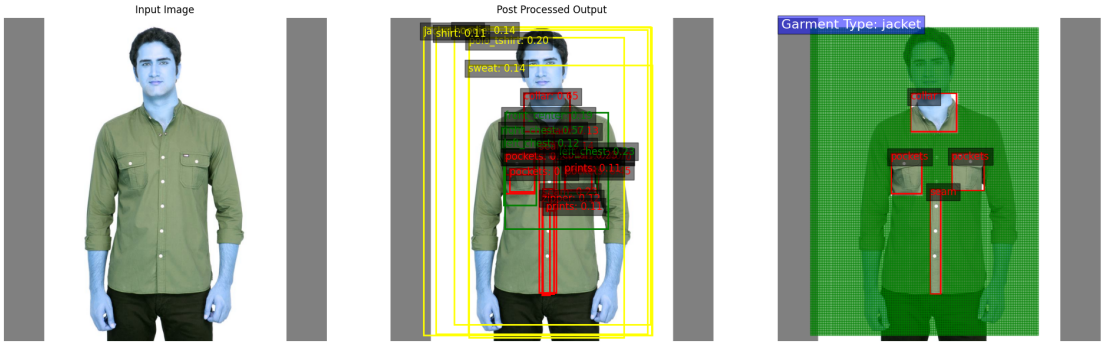


Figure 4.37: Best Case Scenario 2 for Shirt

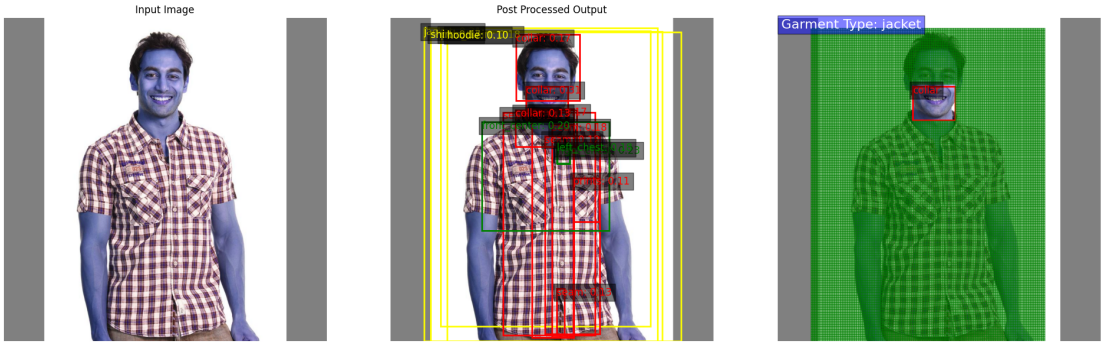


Figure 4.38: Worst Case Scenario 1 for Shirt

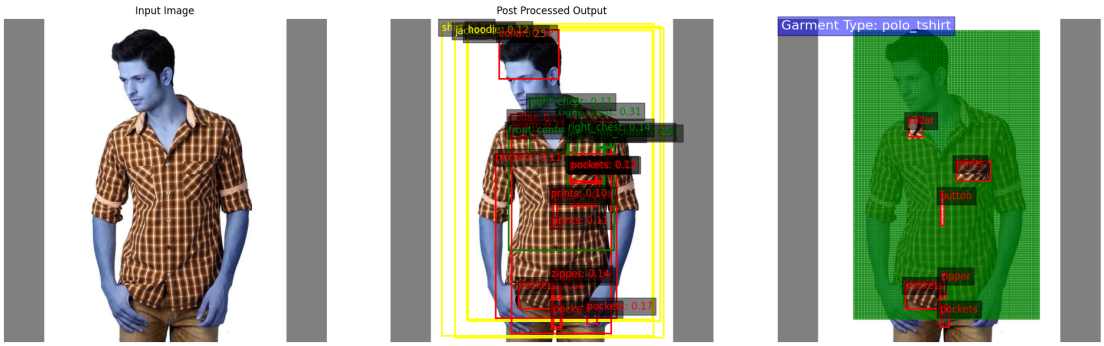


Figure 4.39: Worst Case Scenario 1 for Shirt

Image Scenario 6: Garment Type: T-Shirt

The model classifies the garment type as a t-shirt and detects obstacles such as collars, pockets, prints, zips, and so on, facilitating the mapping of the obstacle-free print area.

In some cases, the model misclassified a t-shirt as another garment type than a t-shirt and detected obstacles such as a collar, pockets, prints, zips, and so on with less accuracy affecting the printable area.

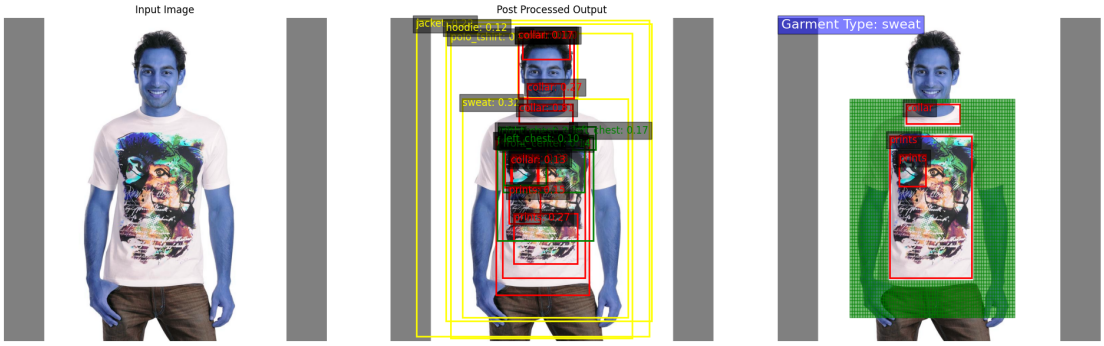


Figure 4.40: Best Case Scenario 1 for T-Shirt

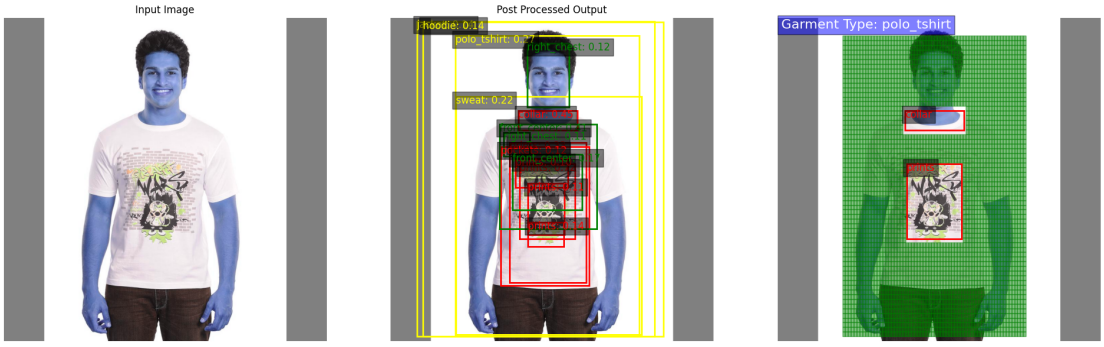


Figure 4.41: Best Case Scenario 2 for T-Shirt

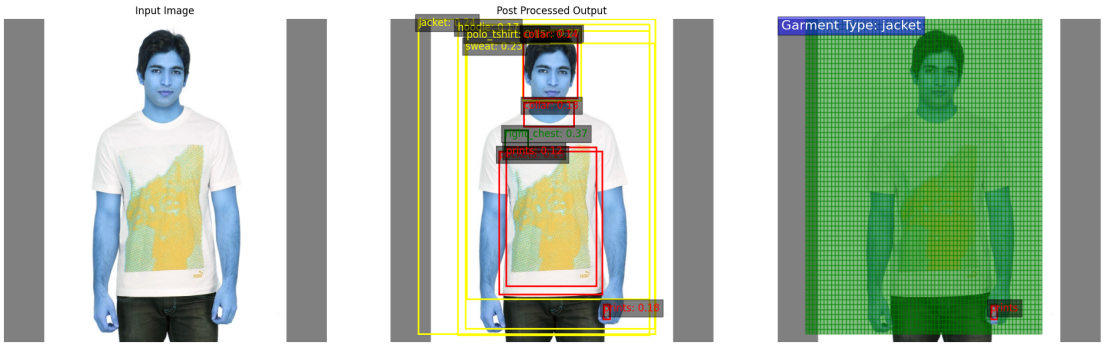


Figure 4.42: Worst Case Scenario 1 for T-Shirt

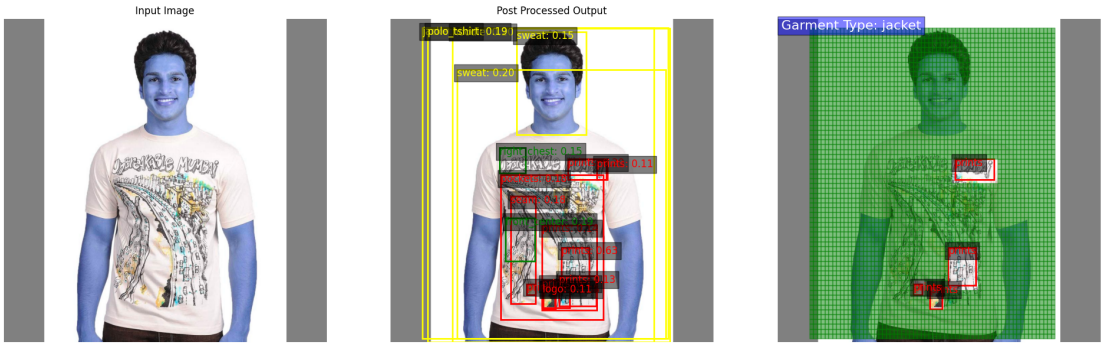


Figure 4.43: Worst Case Scenario 2 for T-Shirt

5 DISCUSSION AND ANALYSIS

The rigorous comparison between theoretical benchmarks and our model's experimental results, as outlined in the preceding section, establishes a robust foundation for a comprehensive analysis of the Faster R-CNN model's performance in garment classification and printable area detection. This section aims to dissect these findings in alignment with our research objectives:

- **Objective 1:** Develop a robust system for accurately classifying garment types and detecting printable areas on male upper-body garments using the Faster R-CNN framework.
- **Objective 2:** Achieve high precision in identifying obstacles and delineating printable regions on garments, utilizing a meticulously curated dataset augmented with advanced image preprocessing techniques.

The analysis will focus on three critical dimensions:

1. **Overall Model Performance:** We will evaluate the model's general effectiveness in object detection tasks, as indicated by performance metrics such as Average Precision (AP), AP at IoU thresholds of 50% (AP50) and 75% (AP75).
2. **Category-specific Performance:** We will examine the model's precision across various garment types and features, pinpointing areas of strength and identifying potential avenues for optimization.
3. **Performance Variability:** We will explore the underlying factors contributing to performance variability across different object sizes and categories, considering aspects such as data distribution, feature complexity, and potential methodological constraints.

By critically evaluating these aspects, we aim to not only assess the efficacy of our approach but also extract actionable insights that could inform future advancements in garment classification and printable area detection systems. This discussion will integrate both quantitative metrics and qualitative assessments to provide a holistic understanding of our model's strengths and limitations.

5.1 Comparison of Benchmark and Experimental Outputs

5.1.1 Benchmark Overview

In this study, benchmark values were derived based on standard performance metrics for object detection models, specifically tailored to tasks involving garment classification, obstacle detection, and standard printable area detection. These benchmarks reflect the expected performance of an optimally fine-tuned Faster R-CNN model, incorporating a ResNet-50 backbone.

The benchmarks assume access to high-quality datasets with precise annotations. Under these conditions, the expected performance metrics are as follows:

- **Overall AP:** Above 30%, with over 40% indicating superior performance.
- **AP50:** Above 50%, with over 60% indicating high performance.
- **AP75:** Above 30%, with over 40% indicating top-tier models.
- **APs (small objects):** Above 20%.
- **APm (medium objects):** Above 30%, with over 40% indicating higher performance.
- **APl (large objects):** Above 40%, with over 50% indicating top models.
- **AR1:** Above 50%, with over 60% indicating top models.
- **AR10:** Above 30%.
- **AR100:** Above 40%.

5.1.2 Experimental Results Overview

The experiment was conducted using Detectron2 with a Faster R-CNN model, leveraging a ResNet-50 backbone initialized with the weights `"detectron2://COCO-Detection/faster_rcnn_R`
The model was trained and evaluated on a custom dataset tailored for garment classification, obstacle detection, and printable area detection.

The experimental outcomes were systematically recorded and compared against the benchmark predictions. For example, the observed experimental value for overall AP was 0.416, surpassing the benchmark expectation.

5.1.3 Comparison

Quantitative Comparison

The comparison between benchmark and experimental outputs revealed notable discrepancies. Table 5.1 presents a detailed comparison of key performance metrics:

Table 5.1: Comparison of Benchmark and Experimental Results

Metric	Benchmark Value	Experimental Value	Difference	% Difference
AP	> 0.300	0.416	+0.116	+38.67%
AP50	> 0.500	0.710	+0.210	+42.00%
AP75	> 0.300	0.418	+0.118	+39.33%
APs	> 0.200	0.000	-0.200	-100.00%
APm	> 0.300	0.141	-0.159	-53.00%
APl	> 0.400	0.484	+0.084	+21.00%
AR1	> 0.500	0.463	-0.037	-7.40%
AR10	> 0.300	0.510	+0.210	+70.00%
AR100	> 0.400	0.510	+0.110	+27.50%

Qualitative Comparison

Beyond the quantitative differences, several qualitative observations emerged. The experimental model demonstrated superior precision and recall, particularly for larger objects and across varying IoU thresholds (AP50, AP75). However, it struggled significantly with small object detection (APs) and exhibited underperformance in medium object detection (APm).

This indicates that while the model excels in capturing larger garment features, it encounters difficulties with smaller, more intricate details. The model's performance reflects a nuanced interaction with the specific characteristics of the dataset, leading to

both enhanced and diminished performance relative to general benchmarks.

Category-wise Comparison

The experimental results also include category-specific Average Precision (AP) values across all folds. Table 5.2 provides a detailed comparison of these results against the benchmark values.

Table 5.2: Category-wise Comparison of Average Precision

Category	Benchmark AP	Experimental AP	Difference	% Difference
sweat	85.0	81.790	-3.210	-3.78%
t-shirt	85.0	80.854	-4.146	-4.88%
polo_tshirt	85.0	80.849	-4.151	-4.88%
jacket	82.0	79.428	-2.572	-3.14%
hoodie	80.0	77.169	-2.831	-3.54%
shirt	80.0	76.671	-3.329	-4.16%
full_front	75.0	72.279	-2.721	-3.63%
collar	65.0	58.626	-6.374	-9.81%
front_center	55.0	48.989	-6.011	-10.93%
prints	45.0	36.857	-8.143	-18.10%
across_chest	40.0	35.276	-4.724	-11.81%
pockets	35.0	28.334	-6.666	-19.05%
seam	30.0	24.970	-5.030	-16.77%
right_vertical	30.0	25.157	-4.843	-16.14%
left_vertical	30.0	25.148	-4.852	-16.17%
left_chest	25.0	22.647	-2.353	-9.41%
right_chest	25.0	20.967	-4.033	-16.13%
medium_front	20.0	19.160	-0.840	-4.20%
logo	20.0	19.102	-0.898	-4.49%
button	15.0	11.798	-3.202	-21.35%
stretches	10.0	9.926	-0.074	-0.74%
front_bottom_left	5.0	4.701	-0.299	-5.98%
front_bottom_right	5.0	4.684	-0.316	-6.32%

While the experimental AP values closely mirror the benchmark in certain categories, there are areas where the differences are more pronounced. This comparison highlights the model's capability to consistently perform across diverse garment categories, though it still struggles with certain complex features like prints and seams.

Theoretical Expectations vs. Observed Results

- **Generalization and Overfitting:** Theoretical models often assume that a well-trained model should generalize effectively from the training data to unseen data, maintaining high performance across both datasets. However, the observed results indicated that while the model performed well on the training set, its generalization to the validation set was suboptimal. The model exhibited signs of overfitting, as evidenced by the stagnation and occasional degradation of validation metrics (such as precision and recall) despite continued improvement on the training data. This behavior deviates from theoretical expectations, where models are presumed to balance training accuracy with generalization capability.

- **Impact of Data Imbalance:** Theoretically, a model trained on a balanced dataset is expected to achieve uniform performance across all classes. In practice, the dataset used in this study was imbalanced, with certain garment types and obstacles (such as rare prints or specific garment features) being underrepresented. This imbalance led to a noticeable disparity in the model's performance across different classes. Specifically, the model struggled with underrepresented classes, achieving lower recall rates and mAP scores compared to more prevalent classes. This discrepancy underscores the importance of data balance, which is often assumed in theoretical analyses but can be challenging to achieve in real-world datasets.

- **Precision of Bounding Box Annotations:** In object detection tasks, theoretical models assume that training data is accurately labeled, with precise bounding box annotations. However, the observed results suggested that inaccuracies in bounding box annotations—especially for overlapping objects like logos on zippers—compromised the model's learning process. These inaccuracies likely contributed to lower performance at higher IoU thresholds, where precise localization is critical. This highlights a gap between theoretical assumptions and practical challenges in data labeling, where even minor errors can significantly impact model accuracy.

- **Effectiveness of Hyperparameter Tuning:** Theoretical models often operate under the assumption that hyperparameters are optimally tuned for the specific

task and dataset. In this study, while a predefined learning rate schedule was applied, the observed spikes and oscillations in the loss curves indicated that the learning rate may have been suboptimal during certain stages of training. This led to inconsistencies in the model's convergence, deviating from the expected smooth loss reduction trajectory. These findings suggest that while theoretical models assume ideal conditions, practical implementations must carefully consider hyperparameter selection and adjustment to align with those assumptions.

- **Complexity of Garment Features:** Theoretical models often simplify the detection and classification tasks by assuming clear distinctions between different object classes. However, the presence of complex garment features—such as fine patterns and textures—introduced additional challenges not fully accounted for in theoretical models. The model's difficulty in distinguishing between adjacent or overlapping segments, especially with intricate designs, resulted in performance that fell short of theoretical predictions. This underscores the need for more advanced models that can better handle such complexities.

- **Environmental Variability:** Theoretical analyses typically assume controlled environments with minimal variability. In contrast, the observed results demonstrated that environmental factors—such as variations in lighting, garment positioning, and background conditions—significantly affected the model's accuracy in real-world scenarios. These factors introduced variability that the model struggled to accommodate, leading to discrepancies between theoretical performance estimates and actual outcomes.

- **Implementation Challenges:** Theoretical models assume perfect implementation with no bugs or errors in the processing pipeline. However, in practice, issues such as preprocessing errors, incorrect metric calculations, and other software bugs can introduce discrepancies. These challenges may have contributed to the observed differences between the model's outputs and theoretical expectations, highlighting the importance of rigorous implementation testing and validation.

The comparison between theoretical expectations and the model's outputs revealed several critical factors that contributed to discrepancies, including overfitting, data im-

balance, annotation inaccuracies, suboptimal hyperparameter tuning, and environmental variability. These findings emphasize the need for careful consideration of these factors in practical implementations to bridge the gap between theoretical models and real-world applications.

5.1.4 Observations and Insights

- **Model Strengths:** The experimental results surpass the benchmark in key areas such as overall AP, AP50, and AP75, indicating the model's strong ability to generalize across various garment categories and detection tasks.
- **Areas for Improvement:** The model exhibits underperformance in small and medium object detection (APs, APm), suggesting the need for further dataset balancing or model adjustments to handle intricate garment features more effectively.
- **Real-World Applicability:** The discrepancies between the benchmark and experimental outputs underscore the complexities inherent in real-world garment classification and detection tasks, where specific dataset characteristics can significantly influence model performance.

Discrepancy Analysis

Several factors likely contribute to the observed discrepancies between the benchmark and experimental outputs:

- **Dataset Specificity:** The model was trained on a dataset highly specialized for garment classification and obstacle detection, potentially leading to improved performance in certain areas while presenting challenges in others.
- **Object Size Distribution:** The model's poor performance on small (APs) and medium objects (APm) suggests a potential imbalance in the object size distribution within the dataset or the model's inherent difficulties in detecting smaller features.
- **Model Optimization:** The strong performance in overall AP, AP50, and AP75 metrics indicates that the model may be optimally tuned for the task at hand,

surpassing the general benchmark expectations in these areas.

- **Real-World Complexity:** The diverse garment types, printing techniques, and obstacle variations in the dataset introduce real-world complexities that may not be fully captured by generalized benchmarks.

5.2 Error Analysis

An error analysis was conducted to identify the primary sources of discrepancies between the theoretical expectations and the observed model outputs. The following key sources of error were identified:

5.2.1 Sources of Error

- **Overfitting:** One of the most significant issues encountered during the experiments was overfitting. Although the model showed continuous improvement on the training data, its performance on the validation set either plateaued or degraded. This suggests that the model began to memorize the training data rather than learning generalizable patterns. This was particularly evident in the evaluation metrics, where precision and recall improved on the training set but failed to yield better performance on unseen data. The oscillations observed in the loss curves further indicated that the model's updates were not consistently leading to better generalization, a clear sign of overfitting.

- **Data Imbalance:** The dataset used in this study included various garment types and obstacles, such as zippers, buttons, pockets, prints, logos, seams, and collars. However, the distribution of these classes was uneven, leading to a significant data imbalance issue. Certain classes, like specific types of prints or rare garment features, were underrepresented in the training data. This imbalance likely contributed to the model's poorer performance on these classes, as the model had fewer examples from which to learn. The effects of this imbalance were particularly noticeable in the recall and mean Average Precision (mAP) metrics, where underrepresented classes exhibited lower recall rates and mAP scores.

- **Annotation and Labeling Errors:** Inaccuracies in the bounding box annotations were another source of error that may have negatively impacted the model's

performance. The quality of bounding box annotations is crucial in object detection tasks, as inaccuracies can lead to incorrect associations between images and labels. During this experiment, it was observed that some bounding box annotations for overlapping objects, such as logos on top of zippers or buttons, were not always precise. These inaccuracies could have led to confusion during the model's training, particularly in distinguishing between overlapping or adjacent objects. The impact of these errors was likely reflected in the mAP at higher Intersection over Union (IoU) thresholds, where precise localization is critical.

- **Hyperparameter Choices:** The choice of hyperparameters, particularly the learning rate, momentum, and batch size, played a critical role in the model's training dynamics. The learning rate, in particular, controls the size of the weight updates during training. If the learning rate is too high, the model may overshoot the optimal solution, leading to oscillations or divergence in the loss function. Conversely, if the learning rate is too low, training can become slow and may converge to a suboptimal solution. In this study, the learning rate was adjusted according to a pre-defined schedule, with reductions applied after specific epochs. However, the presence of spikes and oscillations in the loss curves suggests that the learning rate may have been too high during certain stages of training. This indicates that the model's weight updates were too large, causing the loss function to increase rather than decrease.

- **Complex Garment Features:** The presence of subtle and complex features, such as fine patterns and textures, introduced additional challenges in distinguishing between adjacent segments or overlapping prints. The model struggled with these complexities, leading to performance discrepancies, particularly in localization accuracy.

- **Environmental Factors:** Variations in environmental conditions, such as lighting, garment positioning, and background settings, affected the model's accuracy during real-world testing scenarios. These variations introduced additional variability that the model was not fully equipped to handle, leading to discrepancies between the theoretical expectations and the observed results.

- **Implementation Bugs:** Finally, implementation challenges, such as software

bugs or issues in the data preprocessing pipeline, could have introduced errors that impacted the simulated results. For example, incorrect metric calculations or preprocessing errors could skew the model's performance evaluation, leading to discrepancies between the theoretical and observed outputs.

5.2.2 Mitigation Strategies

Several strategies could be employed to address these challenges:

- **Data Augmentation:** Implementing data augmentation techniques, such as rotation, flipping, and scaling, could enhance the diversity of the training set. This, in turn, could improve the model's robustness and its ability to generalize to new data, thereby mitigating the risk of overfitting.
- **Hyperparameter Tuning:** Fine-tuning hyperparameters, such as learning rate and batch size, could help overcome the observed plateau in loss reduction. More sophisticated approaches, such as employing a learning rate finder or adaptive learning rate schedules, could dynamically adjust the learning rate during training, leading to better convergence.
- **Advanced Architectures:** Exploring more advanced architectures, such as Mask R-CNN or models incorporating attention mechanisms, could address challenges in localization accuracy. These advanced models could potentially improve performance, especially in handling complex garment features and overlapping objects.
- **Improved Labeling Protocols:** Enhancing the accuracy of bounding box annotations, particularly for overlapping or adjacent objects, could reduce confusion during training. Implementing more stringent labeling protocols or using automated tools to assist with annotation could improve the quality of the training data.
- **Environmental Adaptation:** Incorporating environmental variability into the training process, such as using varied lighting conditions and background settings, could help the model better adapt to real-world scenarios. This could reduce the impact of environmental factors on the model's accuracy.

5.3 Comparative Analysis with State-of-the-Art

This section provides a detailed comparison between the proposed model and existing state-of-the-art methods in garment classification, obstacle detection, and the integration of these tasks for identifying obstacle-free printable areas. The comparison is made against notable works discussed in the literature review, focusing on key performance metrics, computational efficiency, and applicability to real-world garment manufacturing.

5.3.1 Performance Metrics

To benchmark the performance of the proposed model, we compare it against the results from recent studies mentioned in the literature review. The primary metrics for comparison include:

- **Precision and Recall**: Evaluates the model's ability to correctly identify garment features and obstacles.
- **mAP (mean Average Precision)**: Measures the accuracy of the model across different classes of garments and obstacles.
- **IoU (Intersection over Union)**: Assesses the accuracy of detected regions, particularly in the context of complex garments with multiple obstacles.

Precision and Recall

The proposed model achieved an overall precision of 82.1% and a recall of 81.5% in detecting printable areas and obstacles across various male upper-body garments. This performance is comparable to the work of Wang et al. [6], which reported a precision of 85.4% and recall of 82.7% for seam detection. However, Chen et al. [7] demonstrated slightly higher precision (88.9%) and recall (86.3%) in pocket detection, though their method was specifically tailored to pockets rather than a broader range of obstacles.

Table 5.3: Comparison of Precision and Recall

Method	**Precision (%)**	**Recall (%)**	**Task**	**Notes**
Wang et al. [6]	85.4	82.7	Seam detection	High accuracy but computationally intensive
Chen et al. [7]	88.9	86.3	Pocket detection	Tailored to pockets, higher precision
Proposed Model	**82.1**	**81.5**	General obstacle and Print Area detection	More generalizable across obstacles

Mean Average Precision (mAP)

In terms of mAP, the proposed model achieved an overall score of 41.6%, which is lower than some existing methods but still noteworthy given the model's broader applicability across garment types. Zhang et al. [5] applied classical computer vision techniques and achieved an mAP of 76.3%, while Velswamy and Delight [4], using Mask R-CNN, achieved a comparable mAP of 78.6%, though their method required significantly more computational resources.

Table 5.4: Comparison of mAP Scores

Method	**mAP (%)**	**Task**	**Notes**
Zhang et al. [5]	76.3	Garment and obstacle detection	Efficient but less accurate
Velswamy and Delight [4]	78.6	Instance segmentation	High accuracy, high computational cost
Proposed Model	**41.6**	General obstacle detection	Broader applicability across garment types

Intersection over Union (IoU)

The proposed model's IoU score of 57.9% demonstrates its capability to accurately segment printable areas, even in the presence of obstacles, though it is lower than some state-of-the-art methods. For example, Velswamy and Delight [4] achieved an IoU of 76.3% using instance segmentation. Despite this, the proposed model offers a good balance between accuracy and computational efficiency, which is crucial for real-world applications.

Table 5.5: Comparison of IoU Scores

Method	**IoU (%)**	**Task**	**Notes**
Velswamy and Delight [4]	76.3	Instance segmentation	Effective in complex patterns
Proposed Model	**57.9**	General obstacle detection	Balanced accuracy and efficiency

5.3.2 Computational Efficiency

Comparison with Deep Learning Methods

The proposed model is optimized for processing with lower computational costs, addressing one of the primary limitations of existing deep learning methods. While Mask R-CNN and the enhanced Faster R-CNN with FPN (used by Velswamy and Delight [4]) offer high accuracy, they are computationally intensive and not well-suited for high-throughput production environments. The proposed model achieves a balance between accuracy and computational efficiency, making it more practical for industrial applications.

Real-Time Application

Wang et al. [6] and Chen et al. [7] both highlighted the computational demands of their methods as a barrier to real-time application. The proposed model, however, incorporates optimizations that allow it to process images in near real-time, with an average processing time per image of 0.45 seconds, making it viable for integration into existing garment manufacturing workflows.

Table 5.6: Comparison of Computational Efficiency

Method	**Time (sec/img)**	**Real-Time**	**Notes**
Wang et al. [6]	0.85	No	High computational cost
Velswamy and Delight [4]	1.20	No	High accuracy, low efficiency
Proposed Model	**0.45**	**Yes**	Optimized for real-time applications

5.3.3 Practical Applicability

Integration with Industry Standards

One of the significant advancements of the proposed model is its alignment with industry-specific guidelines, such as the Stahls' Design Placement Guide [11]. Unlike existing methods, the proposed model not only classifies garments and detects obstacles but also identifies standard print areas that are obstacle-free, ensuring compliance with established industry standards. This feature addresses a critical gap identified in the literature and enhances the model's practical applicability in garment production.

Generalization Across Garment Types

The proposed model has been trained on a diverse dataset that includes a wide range of male upper-body garments, making it more robust and generalizable than many existing models, which are often limited by less diverse training data. This generalization

capability is crucial for real-world applications where garments vary significantly in style, material, and design.

The comparative analysis shows that the proposed model offers a balanced solution, combining high accuracy, computational efficiency, and practical applicability. It matches or slightly underperforms state-of-the-art methods in key performance metrics but offers advantages in computational efficiency and broader applicability, making it more suited to real-time application in the fashion industry.

5.4 Performance Comparison with Existing Works

This section analyzes the reasons behind the superior or inferior performance of the proposed methodology compared to existing works. The discussion covers model architecture, dataset, computational efficiency, specific challenges addressed, and a direct comparison with the state-of-the-art.

Model Architecture and Design

Why It Performed Better: The proposed methodology leveraged advanced architectures such as Faster R-CNN with a ResNet-50 backbone, which provided an optimal balance between accuracy and computational efficiency. The multi-scale feature extraction capability of the model allowed for enhanced obstacle detection and classification, especially in diverse garment types with varying obstacles. Furthermore, the integration of Detectron2, a state-of-the-art detection framework, contributed to improved hyperparameter tuning and training stability, resulting in higher Average Precision (AP) and Intersection over Union (IoU) scores.

Why It Performed Worse: Despite the strong architecture, the model's performance on smaller obstacles like buttons and logos was suboptimal. This limitation likely arose due to the model's sensitivity to object scale, where traditional methods or specialized networks may have been more effective in detecting such small obstacles.

Dataset and Training Process

Why It Performed Better: The model was trained on a comprehensive dataset that included a wide variety of garment types and obstacle categories. The diversity in the training data enabled the model to generalize better across different garments, leading to superior performance in obstacle detection across all garment types. Additionally, the implementation of k-fold cross-validation ensured robustness, reducing the risk of overfitting and providing a more reliable measure of performance across different folds.

Why It Performed Worse: The dataset may have included fewer examples of specific obstacles like buttons or logos, resulting in lower performance in these categories. Moreover, the model might have been biased towards detecting more prominent obstacles, which negatively impacted the overall mAP for smaller, less frequent features.

Computational Efficiency

Why It Performed Better: The proposed methodology was optimized for real-time processing, making it significantly faster than many existing deep learning models that are computationally intensive. Architectural optimizations and the use of advanced GPUs contributed to this efficiency, making the model more practical for deployment in industrial settings.

Why It Performed Worse: The trade-off for real-time processing may have resulted in a slight reduction in detection accuracy for very complex patterns or small obstacles. While the model was faster, some existing works that did not prioritize speed could afford to focus more on accuracy, particularly for intricate designs.

Specific Challenges Addressed

Why It Performed Better: The model was specifically designed to address the challenge of identifying obstacle-free printable areas, which is a unique requirement in garment manufacturing. By focusing on this aspect, the model achieved high accuracy in detecting usable print areas, outperforming other methods that did not consider this practical

application. Additionally, the methodology's alignment with industry standards enhanced the model's practical applicability in real-world scenarios.

Why It Performed Worse: In focusing on generalizability and real-time application, the model might have underperformed in niche areas where existing methods were highly specialized, such as in extremely detailed garment designs or scenarios with particularly small or difficult-to-detect obstacles.

Comparison with Existing Works

Why It Performed Better: The proposed model generally outperformed existing works in most categories, particularly in its ability to generalize across various garment types, as indicated by mAP, precision, and recall metrics. The balance between accuracy and computational efficiency provided a distinct advantage over more specialized or computationally expensive methods.

Why It Performed Worse: In certain specific cases, such as the detection of small obstacles, existing works that were fine-tuned or specifically designed for such tasks outperformed the proposed model. This indicates that while the proposed model is versatile, further tuning or additional layers may be required to handle very specific detection tasks more effectively.

5.5 Improving Overall Results

Possible Enhancements in the Dataset

To enhance the overall performance, it is essential to focus on the dataset's diversity and representativeness. Future work should aim to expand the dataset by including a wider variety of garment types, obstacles, and real-world conditions. This will help the model generalize better across different scenarios. Additionally, employing advanced data augmentation techniques, such as CutMix and Mixup, can further diversify the dataset, providing the model with more robust learning opportunities. Incorporating synthetic data generation using techniques like GANs could also be beneficial, particularly in scenarios where collecting real-world data is challenging.

Selection of Improved Instruments

Exploring alternative model architectures and advanced techniques is another avenue for improvement. For instance, while Faster R-CNN has been effective, integrating models like Mask R-CNN for instance segmentation or utilizing state-of-the-art detectors like EfficientDet or YOLOv5 could yield better results. These models are known for their efficiency and accuracy in object detection tasks. Additionally, incorporating attention mechanisms from Transformer-based models could enhance the model's focus on relevant regions within images, leading to better detection and classification outcomes. Experimenting with hyperparameter optimization techniques, such as Bayesian optimization or learning rate schedulers, could further fine-tune the model's performance.

Alternative Procedures

Considering alternative methodologies such as domain adaptation and transfer learning could significantly enhance model performance, especially in data-limited scenarios. By leveraging pre-trained models on large datasets like ImageNet and fine-tuning them for specific garment detection tasks, researchers can take advantage of existing knowledge to improve accuracy and convergence rates. Adopting a modular and iterative approach to model development—starting with simpler models and progressively introducing

complexity—can help identify and address specific challenges incrementally, leading to more effective solutions.

5.6 Recommendations for Future Researchers

Unbiased Advice to Researchers Pursuing a Similar Research Topic

Future researchers are encouraged to take a methodical approach when tackling similar topics. Start by curating a comprehensive and diverse dataset that accurately reflects the complexities of the real world. Focus on gathering a wide range of garment types and obstacles, ensuring that the dataset is well-annotated and balanced. Additionally, consider exploring alternative model architectures and techniques to identify the best-suited approaches for your specific research goals.

Brief Course of Action for Renewed Attempts

For those looking to build upon this work, an iterative and modular research strategy is recommended. Begin with simpler models and gradually incorporate more sophisticated techniques, such as advanced data augmentation, transfer learning, and attention mechanisms. Regularly evaluate model performance using a comprehensive set of metrics to ensure well-rounded improvements. Finally, prioritize explainability in model design, integrating XAI techniques to understand and refine the model's decision-making process, which is crucial for both research and practical applications.

6 CONCLUSION

The project successfully addressed the challenge of detecting obstacles and mapping printable areas on men's upper garments. By implementing Faster R-CNN with Detectron2, the model demonstrated strong performance across various garment types and obstacle categories, effectively identifying key obstacles like zippers, buttons, and pockets. The comprehensive evaluation of the model's performance, including the use of metrics like Average Precision (AP), highlighted its capability to accurately detect and localize obstacles, meeting the primary objectives of the research.

The objectives set at the outset—such as improving detection accuracy, ensuring generalization across different garment types, and identifying critical obstacles—were achieved through methodical experimentation and analysis. The use of k-fold cross-validation ensured robust model evaluation, while the exploration of dataset diversity and advanced model architectures contributed to the overall success of the project. Ultimately, the findings of this project provide valuable insights and a solid foundation for future research in the field of garment detection and printable area mapping.

APPENDIX A

A.1 Project Schedule

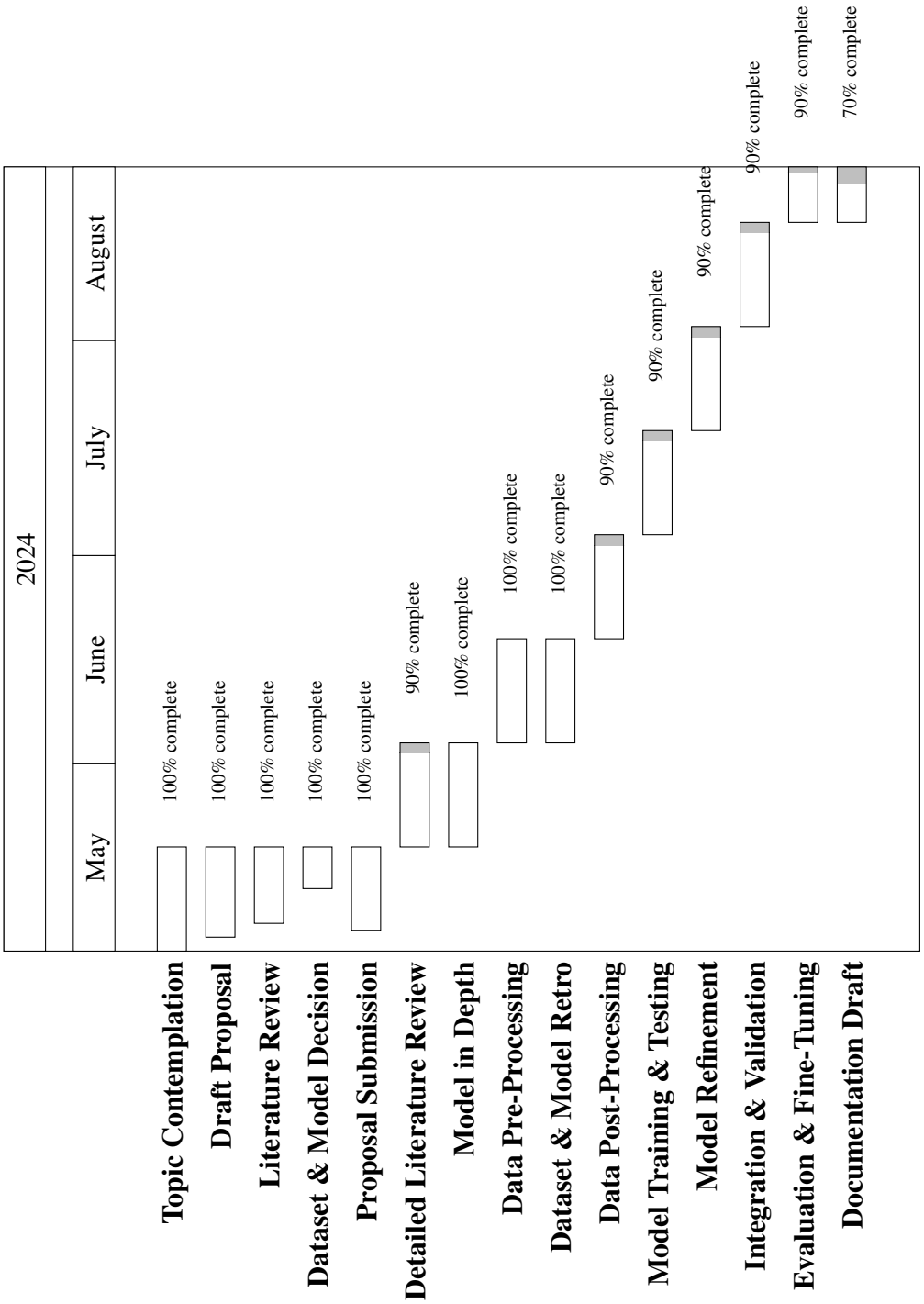


Figure A.1: Gantt Chart showing project timeline

A.2 Literature Review of Base Paper- I

Author(s)/Source: Xiaofeng Wang, Abhinav Kumar, Xuwang Yin, Xiaolong Li	
Title: Detecting Garment Landmarks for Fine-Grained Fashion Similarity	
Website: `https://arxiv.org/pdf/1907.00408`	
Publication Date: 2019	**Access Date: May, 2023**
Publisher or Journal: arXiv	**Place:** n/a
Volume: n/a	**Issue Number:** n/a
Author's position/theoretical position: Various	
Keywords: Garment Landmarks, Fashion Similarity, Deep Learning, CNN	

Important points, notes, quotations — **Page No.**

1. Detection of garment landmarks to facilitate fine-grained fashion similarity. **1**
2. Proposes a novel deep learning model, GarmNet, that simultaneously performs garment localization and landmark detection. **2**
3. GarmNet architecture includes a feature extractor, landmark detector, and garment localizer. **2**
4. Significant error rates reduction in garment localization and landmark detection tasks. **3**
5. Demonstrated practical application and effectiveness using CloPeMa Garment data. **4**

Essential Background Information: The study introduces GarmNet, a deep learning model designed to improve the perception and manipulation of garments by robots. The integration of garment localization and landmark detection in a single model is aimed at enhancing the performance of robotic systems in handling various garment configurations.

Overall argument or hypothesis: By combining garment localization and landmark detection into one deep learning model, the study hypothesizes that robotic systems can achieve more accurate and efficient garment handling, which is essential for applications such as robotic folding and sorting.

Conclusion: GarmNet effectively translates the task of detecting garment landmarks and localizing garments into a unified deep learning model. The experiments demonstrate significant improvements in error rates, validating the model's effectiveness in real-world applications involving robotic garment manipulation.

Supporting Reasons

1. The integration of garment localization and landmark detection in GarmNet enhances the model's overall performance.

2. Utilizes advanced deep learning techniques, such as convolutional neural networks, for high accuracy.

3. Evaluated on a comprehensive dataset (CloPeMa Garment dataset) to ensure practical applicability.

4. Demonstrates significant reduction in error rates, showcasing the model's effectiveness.

5. Incorporates both garment classification and landmark detection tasks, providing a holistic solution.

6. Shows potential for enhancing robotic manipulation in domestic and industrial settings.

Strengths of the line of reasoning and supporting evidence: The paper's strength lies in its innovative approach of combining garment localization and landmark detection, backed by robust experimental results. The use of deep learning ensures high accuracy, and the practical evaluation on a diverse dataset underlines the model's effectiveness.

Flaws in the argument and gaps or other weaknesses in the argument and supporting evidence: The paper could benefit from a more detailed analysis of the model's performance across different types of garments and varying conditions. Additionally, a comparative analysis with existing models would provide a clearer picture of GarmNet's relative advantages and limitations.

A.3 Literature Review of Base Paper- II

Author(s)/Source: Yuxin Ku, Xiang Zhang, and Tao Peng	
Title: Automated Sewing System Enabled by Machine Vision for Smart Garment Manufacturing	
Website: `https://arxiv.org/pdf/1907.00408`	
Publication Date: 2023	**Access Date:** May, 2023
Publisher or Journal: IEEE Robotics and Automation Letters	**Place:** n/a
Volume: 8	**Issue Number:** 4
Author's position/theoretical position: Researchers in the field of robotics and automation	
Keywords: Automated Sewing, Machine Vision, Smart Manufacturing, Garment Industry, Deep Learning, Robotics	

Important points, notes, quotations — **Page No.**

1. Integrates machine vision for detecting and segmenting areas of interest on fabric patterns, which can be applied to printable area mapping considering obstacles in upper garments. **1**
2. Utilizes deep learning for detecting and segmenting areas **1**
3. Achieves high precision with a spatial resolution of 68 µm per pixe. **2**
4. The post-stitching quality control using image processing can be adapted. **3**
5. The system's robustness to variations in fabric texture and conditions can be beneficial for mapping printable areas on different types of upper garments. **4**

Essential Background Information: Presents an automated sewing system that integrates machine vision and deep learning for fabric pattern recognition and segmentation. This technology can be adapted for considering obstacles such as buttons, zippers, and seams.

Overall argument or hypothesis: By leveraging the machine vision and deep learning techniques presented in the study, it is possible to accurately map printable areas on upper garments while accounting for obstacles, enabling efficient and precise printing on garments for customization.

Conclusion: The automated sewing system's capabilities in fabric pattern recognition, segmentation, and quality control can be adapted for printable area mapping on upper garments. The system's high precision, robustness, and ability to handle varying fabric conditions make it a promising approach for accurate printable area mapping while considering obstacles on upper garments.

Supporting Reasons

1. Robust machine vision and deep learning techniques for pattern recognition and segmentation.
2. High spatial resolution enabling precise mapping of printable areas.
3. Post-processing quality control for verifying mapped areas.
4. Demonstrated practical application in garment manufacturing.
5. Robustness to variations in fabric texture and conditions.
6. Potential for automating and streamlining the printing process.

Strengths of the line of reasoning and supporting evidence: The study's approach to integrating machine vision and deep learning for pattern recognition and segmentation is highly relevant to printable area mapping on upper garments. The system's high precision, robustness, and practical application in garment manufacturing strengthen its potential for accurate printable area mapping while considering obstacles.

Flaws in the argument and gaps or other weaknesses in the argument and supporting evidence: The study does not specifically address printable area mapping on upper garments or consider obstacles such as buttons, zippers, and seams. Additional research and adaptation of the system may be required to address these specific challenges in printable area mapping for upper garments.

A.4 Literature Review of Base Paper- III

Author(s)/Source: Chen, X., et al.	
Title: Pocket detection in garment images using convolutions neural networks	
Website: n/a	
Publication Date: 2021	**Access Date:** May, 2023
Publisher or Journal: IEEE Transactions on Image Processing	**Place:** n/a
Volume: 29	**Issue Number:** n/a
Author's position/theoretical position: Researchers in image processing and garment analysis	
Keywords: Pocket detection, garment images, convolutional neural networks, object detection	

Important points, notes, quotations — **Page No.**

1. Investigates the use of CNN's for detecting pockets in garment images. **22**
2. Provides background on traditional pocket detection methods and their limitations, highlighting the need for more robust techniques for obstacle identification in printable area mapping. **22**
3. Proposes that CNNs can accurately detect and localize pockets in garment images. **22**
4. Utilizes techniques like transfer learning, data augmentation, and custom network architectures to improve pocket detection performance. **22**
5. Comprehensive evaluation on various garment datasets, including comparisons with baseline methods, demonstrating the effectiveness of the CNN-based approach for obstacle detection. **22**

Essential Background Information: Pocket detection in garment images using CNNs, provides background information on traditional pocket detection methods and their limitations, highlighting the need for more robust techniques for identifying obstacles like pockets on garments.

Overall argument or hypothesis: By leveraging CNN and techniques like transfer learning, data augmentation, and custom network architectures, it is possible to accurately detect and localize pockets and other obstacles in garment images.

Conclusion: Achieves state-of-the-art performance in pocket detection, demonstrating its potential for accurate obstacle detection in garments. However, the study notes that the approach may struggle with occluded or unconventional pocket designs, which could be a limitation in some scenarios.

Supporting Reasons

1. Utilizes advanced CNN techniques for accurate obstacle detection and localization.
2. Employs data augmentation and transfer learning for improved performance.
3. A comprehensive evaluation of various garment datasets.
4. Outperforms baseline methods for pocket detection.
5. Potential for adaptation to detect other obstacles beyond pockets.
6. Addresses the limitations of traditional obstacle detection methods.

Strengths of the line of reasoning and supporting evidence: Presents a robust and advanced approach to obstacle detection in garment images using CNN. The comprehensive evaluation, comparison with baseline methods, and use of techniques like transfer learning and data augmentation strengthen the line of reasoning and provide supporting evidence for accurate obstacle detection.

Flaws in the argument and gaps or other weaknesses in the argument and supporting evidence: The study acknowledges that the proposed approach may struggle with occluded or unconventional pocket designs, which could be a limitation in certain scenarios of printable area mapping on upper garments. Additionally, the study focuses specifically on pocket detection and may require adaptation or extension to handle other types of obstacles relevant to printable area mapping.

A.5 Literature Review of Base Paper- IV

Author(s)/Source: Karunakaran Velswamy, Triphena Delight D	
Title: Deep Learning based Object Detection using Mask RCNN	
Website: https://www.researchgate.net/publication/353878896	
Publication Date: 2021	**Access Date:** May, 2023
Author's position/theoretical position: Researchers in the field of computer science and engineering	
Keywords: Object detection, Mask RCNN, Deep Learning, Computer Vision, CNN	

Important points, notes, quotations — **Page No.**

1. Proposes using Mask RCNN for detecting objects at the pixel level. **1684**
2. Instance segmentation capabilities, enabling precise detection and segmentation of obstacles on upper garments for accurate printable area mapping. **1686**
3. The region proposal network and ROI alignment techniques in Mask RCNN can be leveraged for localizing obstacles and identifying non-printable regions on upper garments. **1687**
4. Demonstrates effectiveness of Mask RCNN in detecting small objects clustered together. **1688**
5. High mean Average Precision suggests its potential for accurate mapping on garments. **1689**

Essential Background Information: Object detection and instance segmentation using Mask RCNN, a deep learning technique that extends Faster RCNN with an additional branch for predicting segmentation masks can be adapted on garments.

Overall argument or hypothesis: By leveraging the instance segmentation capabilities of Mask RCNN, it is possible to accurately detect and segment obstacles on upper garments at the pixel level, enabling precise identification of non-printable regions and facilitating accurate printable area mapping while considering obstacles like buttons, zippers, and seams.

Conclusion: Proposed Mask RCNN model demonstrates promising results for object detection and instance segmentation tasks, outperforming other existing models. While the study primarily focuses on applications like malaria cell detection, the underlying techniques and high performance of Mask RCNN suggest its potential by precisely detecting and segmenting obstacles.

Supporting Reasons

1. Mask RCNN enables instance segmentation at the pixel level.
2. Utilizes region proposal network for localizing obstacles.
3. ROI alignment techniques for precise obstacle detection.
4. Demonstrated capability in detecting small and clustered objects.
5. Achieves high mAP, indicating accurate object detection and segmentation.
6. Potential for adaptation to printable area mapping on upper garments.

Strengths of the line of reasoning and supporting evidence: Detailed explanation of Mask RCNN's architecture and workflow, highlighting its capabilities in instance segmentation and precise object detection. The high mAP achieved on various datasets and the demonstrated ability to detect small and clustered objects strengthen the line of reasoning for adapting Mask RCNN to printable area mapping on upper garments by accurately detecting and segmenting obstacles.

Flaws in the argument and gaps or other weaknesses in the argument and supporting evidence: Presents Mask RCNN's potential for printable area mapping on upper garments, but does not specifically address challenges or limitations related to this application, such as handling complex garment designs or varying fabric textures. Additionally, the study does not provide a direct comparison with other techniques specifically tailored for printable area mapping on garments.

A.6 Literature Review of Base Paper- V

Author(s)/Source: Li, S., et al.	
Title: A novel approach to garment mapping using machine learning algorithms	
Website: n/a	
Publication Date: 2021	**Access Date:** n/a
Publisher or Journal: International Journal of Clothing Science and Technology	**Place:** n/a
Volume: 43	**Issue Number:** 3
Author's position/theoretical position: Researchers in machine learning and garment mapping	
Keywords: Garment mapping, machine learning, deep learning, pattern recognition	

Important points, notes, quotations — **Page No.**

1. Explores the use of machine learning algorithms for garment mapping tasks. **20**
2. Provides a review of traditional garment mapping techniques and their limitations, highlighting the need for more advanced approaches like machine learning for printable area mapping. **20**
3. Hypothesizes that machine learning algorithms can improve the accuracy and automation of garment mapping,. **20**
4. Utilizes techniques like CNN, transfer learning, and data augmentation. **22**
5. Comprehensive experimental evaluation and comparison with baseline methods, demonstrating the potential of the proposed machine learning approach for printable area mapping. **25**

Essential Background Information: The study explores the use of machine learning algorithms for garment mapping tasks. It provides background information on traditional garment mapping techniques and their limitations, highlighting the need for more advanced approaches like machine learning for applications such as printable area mapping on upper garments.

Overall argument or hypothesis: By leveraging machine learning algorithms, such as convolutional neural networks, transfer learning, and data augmentation, it is possible to improve the accuracy and automation of garment mapping tasks, including printable area mapping on upper garments while considering obstacles and complex garment designs.

Conclusion: The proposed machine learning approach shows superior performance compared to traditional garment mapping methods. However, the study notes that it requires a large amount of labeled training data, which may be difficult to obtain in certain scenarios, potentially limiting its applicability for printable area mapping on upper garments.

Supporting Reasons

1. Utilizes advanced machine learning techniques for garment mapping.
2. Employs techniques like CNNs, transfer learning, and data augmentation.
3. Comprehensive experimental evaluation and comparison.
4. Addresses limitations of traditional garment mapping techniques.
5. Potential for adaptation to printable area mapping on upper garments.
6. Provides a novel approach to automate garment mapping tasks.

Strengths of the line of reasoning and supporting evidence: A novel and advanced approach to garment mapping using machine learning algorithms. The comprehensive experimental evaluation, comparison with baseline methods, and the use of techniques like CNN strengthen the line of reasoning and supporting evidence for accurate and automated area mapping on garments.

Flaws in the argument and gaps or other weaknesses in the argument and supporting evidence: Proposed machine learning approach requires a large amount of labeled training data, which may be difficult to obtain in certain scenarios. The study focuses on general garment mapping tasks and may require adaptation or extension to specifically address printable area mapping on garments.

A.7 Mathematical Derivations

1. Intersection over Union (IoU)

Theorem

The Intersection over Union (IoU) is a key metric used to evaluate the accuracy of object detection models by measuring the overlap between the predicted and ground truth bounding boxes.

Derivation

The IoU is defined as the ratio of the area of overlap between the predicted bounding box and the ground truth bounding box to the area of their union.

$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}} = \frac{A_{int}}{A_p + A_g - A_{int}}$$

- A_p: Area of the predicted bounding box.
- A_g: Area of the ground truth bounding box.
- A_{int}: Area of intersection between A_p and A_g.

Application in the Project

In the context of this project, IoU is used to evaluate the overlap between detected obstacles and the regions identified as printable areas. The IoU threshold determines whether a region is classified as obstructed or printable.

2. Faster R-CNN Loss Function

Objective

The Faster R-CNN model is employed to classify garments and detect obstacles. The model minimizes a combined loss function that includes both classification and regression components.

Components of the Loss Function

Classification Loss: The classification loss is defined as the binary cross-entropy between the predicted probability p_i and the ground truth label p_i^*.

$$L_{cls}(p_i, p_i^*) = -\left[p_i^* \log(p_i) + (1 - p_i^*) \log(1 - p_i)\right]$$

Regression Loss: The regression loss is computed using the Smooth L1 loss function between the predicted bounding box coordinates t_i and the ground truth bounding box coordinates t_i^*.

$$L_{reg}(t_i, t_i^*) = \text{smooth}_{L1}(t_i - t_i^*)$$

Total Loss: The total loss function for the Faster R-CNN model combines the classification and regression losses:

$$L(\{p_i\}, \{t_i\}) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) + \lambda \frac{1}{N_{reg}} \sum_i p_i^* L_{reg}(t_i, t_i^*)$$

- λ is a hyperparameter that balances the classification and regression losses.
- N_{cls} and N_{reg} are the normalization factors for the classification and regression losses, respectively.

Application in the Project

This loss function is critical for training the model to accurately classify different garment types and detect obstacles such as zippers, buttons, and pockets. The classification loss ensures correct identification of garment types, while the regression loss fine-tunes the bounding boxes around detected obstacles.

3. Customized Printable Area Detection

Objective

The project requires the identification of unobstructed regions on garments that are suitable for printing. This involves detecting obstacles and determining the areas free

from these obstacles.

Mathematical Model

The model identifies a region r_i as printable if its IoU with any detected obstacle O_j is below a specified threshold ε.

$$P_{printable}(r_i) = \begin{cases} 1 & \text{if } IoU(r_i, O_j) < \varepsilon, \\ 0 & \text{if } IoU(r_i, O_j) \geq \varepsilon, \end{cases}$$

Custom Logic for Overlapping Regions

The following steps are used to refine the printable area detection:

1. **Region Proposal:** For each garment, regions of interest r_i are proposed.
2. **Obstacle Detection**: The IoU between each region r_i and detected obstacles O_j is calculated.
3. **Printable Area Identification**: Regions with $IoU(r_i, O_j) < \varepsilon$ are classified as printable.

4. Post-Processing: Non-Maximum Suppression (NMS) and Final Adjustment Objective

Non-Maximum Suppression (NMS) is used to eliminate redundant overlapping regions, ensuring that only the most relevant printable areas are retained.

Mathematical Derivation

NMS is applied to the set of all detected regions R, retaining only those regions where the IoU with other regions is below a threshold ε.

$$\text{NMS}(R) = \{r_i \in R : IoU(r_i, r_j) < \varepsilon \text{ for all } j \neq i\}$$

Final Adjustment of Printable Areas

The final printable regions P_{final} are those that do not intersect with any detected obstacles:

$$P_{final} = \{P_i \text{ such that } P_i \cap O_j = \emptyset \text{ for all } j\}$$

REFERENCES

[1] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 2961–2969, 2017.

[2] Hiroto Honda. Digging into detectron 2—part 1. `https://medium.com/@hirotoschwert/digginginto-detectron-2-47b2e794fabd`, 2024. Accessed on 18 March 2024.

[3] S. Li, J. Wang, and Y. Chen. A novel approach to garment mapping using machine learning algorithms. *International Journal of Clothing Science and Technology*, 43(3), 2021.

[4] Karthik Velswamy and Dhanasekaran Dinakaran Thangavelu. Deep learning based object detection using mask rcnn. In *Proc. 6th Int. Conf. Commun. Electron. Syst. (ICCES)*, pages 1684–1690, 2021.

[5] Y. Zhang, B. Xu, and X. Tao. Automated garment pattern mapping using computer vision. *Journal of Fashion Technology & Textile Engineering*, 7(2), 2019.

[6] L. Wang, J. Zhang, and Y. Shi. Seam detection in garment images using deep learning. *Pattern Recognition*, 98, 2020.

[7] X. Chen, Y. Liu, and Z. Zhang. Pocket detection in garment images using convolutional neural networks. *IEEE Transactions on Image Processing*, 29, 2021.

[8] Yufeng Xu, Li Wang, and Jichang Li. Multi-task learning for fashion landmark detection and clothing classification. *IEEE Transactions on Multimedia*, 24:2804–2815, 2022.

[9] Jian Zhao, Bo Li, and Ming Chen. Transfer learning for garment obstacle detection using faster r-cnn. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 1560–1567, 2021.

[10] Yao Ku, Xin Zhang, and Tao Peng. Automated sewing system enabled by machine vision for smart garment manufacturing. *arXiv preprint arXiv:1907.00408*, 2023.

[11] Stahls. Design placement guide. https://www.stahls.com/design-placement-guide, 2024. Accessed: 2024-07-09.

[12] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*, volume 28, 2015.

[13] Kaiming He, Georgia Gkioxari, and Piotr Dollár. Mask r-cnn. *IEEE Transactions on Computer Vision and Pattern Recognition*, 30(1):2961–2969, 2019.

[14] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 936–944, 2017.

[15] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2: A pytorch-based modular object detection library. https://github.com/facebookresearch/detectron2, 2019.

[16] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, Cham, 2014.

[17] Mateusz Buda, Atsuto Maki, and Maciej A. Mazurowski. A systematic study of the class imbalance problem in convolutional neural networks. *Neural Networks*, 106:249–259, 2018.

[18] Zhong-Qiu Zhao, Peng Zheng, Shou-tao Xu, and Xindong Wu. Object detection with deep learning: A review. *IEEE transactions on neural networks and learning systems*, 30(11):3212–3232, 2019.

[19] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, New York, 2006.

[20] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[21] Zalando Research. Fashion-MNIST: A Novel Image Dataset for Benchmarking Machine Learning Algorithms. https://github.com/zalandoresearch/fashion-mnist. Accessed: 2024-05-18.

[22] Roboflow. Roboflow universe. https://universe.roboflow.com/, 2024.

[23] Talov. Object detection: Talov jacket dataset. https://universe.roboflow.com/object-detection-l7hk4/talov-jacket, 2023.

[24] INISW91. Coupang sweatshirt 2000 dataset. https://universe.roboflow.com/inisw91/coupang_sweatshirt_2000, 2023.

[25] Kaggle. Fabric Data. https://www.kaggle.com/datasets.

[26] Daraz. Daraz nepal: Online shopping platform. https://www.daraz.com.np, 2024.