IMPROVING DETECTION OF SIMILARLY DESIGNED RETAIL PRODUCTS WITH SIZE VARIANTS

Conduct by:

Name : Ganendra Raditya P. S.

Student ID : 21523085

PROGRAM STUDI INFORMATIKA – PROGRAM SARJANA

FAKULTAS TEKNOLOGI INDUSTRI

UNIVERSITAS ISLAM INDONESIA

2026

SUPERVISOR ENDORSEMENT PAGE

IMPROVING DETECTION OF SIMILARLY DESIGNED RETAIL PRODUCTS WITH SIZE VARIANTS

THESIS

Conduct By:

Name : Ganendra Raditya P. S.

Student ID : 21523085

Yogyakarta, January 9th, 2026

Advisor,

(Ir. Dhomas Hatta Fudholi, S.T., M.Eng., Ph.D., IPM., ASEAN Eng.)

EXAMINER ENDORSEMENT PAGE

IMPROVING DETECTION OF SIMILARLY DESIGNED RETAIL PRODUCTS WITH SIZE VARIANTS

THESIS

Has been defended in front of the examiners as one of the requirements to obtain a Bachelor of Informatics degree from the Undergraduate Program in Informatics at the Faculty of Industrial Technology, Universitas Islam Indonesia

Yogyakarta, January 9th, 2026

Chair

Ir. Dhomas Hatta Fudholi, S.T., M.Eng., Ph.D., IPM., ASEAN Eng.

A.n Hari Setiaji

Examiner 1

Dr. Syarif Hidayat, S.Kom., M.I.T.

Examiner 2

Chanifah Indah Ratnasari, S.Kom., M.Kom.

Acknowledged by,

Head of Undergraduate Program in Informatics

Faculty of Industrial Technology

Universitas Islam Indonesia

UNIVERSITAS ISLAM INDONESIA * YOGYAKARTA * FAKULTAS TEKNOLOGI INDUSTRI

(Ir. Dhomas Hatta Fudholi, S.T., M.Eng., Ph.D., IPM., ASEAN Eng.)

AUTHENTICITY STATEMENT

The undersigned:

Name	:	Ganendra Raditya P. S.
Student ID	:	21523085

Final project with title:

IMPROVING DETECTION OF SIMILARLY DESIGNED RETAIL PRODUCTS WITH SIZE VARIANTS

Stating that all components and contents in this final project are my own work. If in the future it is proven that some parts of this work are not my own work, the final project submitted as my own work is ready to be withdrawn and ready to bear any risks and consequences.

Thus this statement letter is made, hopefully it can be used properly.

Yogyakarta, January 9th, 2026

(Ganendra Raditya P. S.)

DEDICATION

Everything is devoted to the Creator, the Lord of the worlds. I am grateful to God the Almighty for His guidance and blessings in every step of this journey. I also dedicate this work to my family for their endless support, prayers, and encouragement throughout my studies. Finally, I dedicate this work to all who have supported and inspired me, especially my friends and mentors, whose motivation has been invaluable along the way.

MOTTO

We think a flower on a cliff is beautiful
because we stop our feet at the cliff's edge,
unable to step out into the sky
like that fearless flower.
— Aizen, Bleach

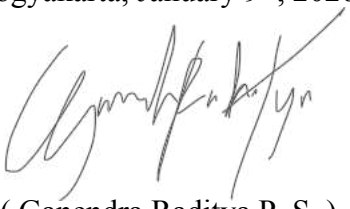
FOREWORD

Alhamdulillah, all praises to Allah Swt. I am grateful for the opportunity to work on this final project. This project reflects a series of learning experiences, challenges, and explorations. Sincerely, I would like to express my gratitude to:

1. Allah Swt., for everything.
2. My family, for their endless support, prayers, and encouragement that have been my foundation throughout my whole life.
3. My supervisor, Mr. Ir. Dhomas Hatta Fudholi, S.T., M.Eng., Ph.D., IPM., ASEAN Eng., for his guidance, valuable advice, and insightful feedback that have helped me navigate this research.
4. All lecturers and staffs in the Informatics Department of UII, for their dedication throughout my study.
5. Every countless individual, especially friends, colleagues, and my extended family for their motivation, discussions, and support, which have made this journey more meaningful and enjoyable.

Although this research is not without its shortcomings, I hope this project can provide value not only for my personal learning but also for anyone who may benefit from it. Lastly, may God grant you all with guidance, blessings, and success in every endeavor. Amen.

Yogyakarta, January 9th, 2026



(Ganendra Raditya P. S.)

ABSTRACT

Retail environments require accurate and consistent shelf monitoring to support inventory management and planogram compliance. However, manual inspection remains inefficient and error-prone, particularly in modern grocery stores that contain thousands of Stock Keeping Units (SKUs). As a result, automated vision-based monitoring systems have been increasingly adopted to replace manual shelf inspection and improve scalability. Moreover, a persistent challenge in automated retail monitoring systems arises when product variants from the same brand share nearly identical packaging designs and differ only in weight or grammage. In such fine-grained scenarios, object detection models frequently confuse product variants, leading to unreliable inventory data and missed detections. This research addresses the problem of fine-grained retail product detection under real-world constraints, where data scarcity and class imbalance limit the effectiveness of deep learning models. To mitigate these challenges, an optimization-driven detection pipeline is proposed, integrating systematic hyperparameter tuning, dataset-level context-aware augmentation, and ensemble model post-processing. Two complementary object detection architectures, YOLOv12 and Faster R-CNN are employed to capture diverse feature representations. Model training is optimized using hierarchical hyperparameter tuning with the Optuna Tree-Structured Parzen Estimator (TPE), while data limitations are addressed through Context-Aware Copy-Paste augmentation. Furthermore, prediction outputs are fused using Weighted Boxes Fusion (WBF) to improve detection consistency. Experimental evaluation on real-world retail shelf images demonstrates that the proposed pipeline reduces error rates compared to baseline detection configurations. Analysis using confusion matrices shows a reduction in misclassification among product variants. These findings indicate that combining architecture diversity, systematic optimization, data-centric augmentation, and ensemble model provides a solution for fine-grained retail product detection in data-limited retail environments.

Keywords: Data augmentation; Ensemble model; Fine-grained object detection; Hyperparameter tuning; Retail product.

GLOSSARY

Augmentation	a strategy to artificially increase the diversity of the training dataset by applying random transformations to improve model generalization.
Backbone	the primary feature extraction network within an object detection architecture responsible for identifying visual patterns from images.
Bounding Box	a rectangular region defined by coordinates (x, y, width, height) that encloses a detected object within an image.
Epoch	one complete pass of the entire training dataset through the algorithm.
Fine-Grained	a classification setting in which the differences between classes are subtle, often involving minor variations in attributes such as small textual details, while the overall visual appearance remains similar.
Grammage	a specific retail term referring to the weight or mass specification of a product.
Ground truth	the factual data used as the standard benchmark to train and evaluate the model.
Hyperparameter	a defined setting of modelling before training starts, controlling how the model learns.
Overfit	a modelling problem where the model learns the training data too closely, including its noise, negatively impacting its performance.
Post-processing	the final stage of the pipeline where predictions are refined into results.
Preprocessing	the initial phase of preparing data to meet the specific requirements.
SKU	a unique identifier used by retailers to track inventory, referring to a specific distinct product variant.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I
INTRODUCTION

1.1 Study Background and Motivation

The global retail industry continues to operate in an environment of immense scale and complexity, with the global market expected to be valued at approximately USD 35.2 trillion in 2025 and projected to exceed USD 50 trillion by 2030 (Bizplanr Team, 2025). Within this highly competitive landscape, inventory management and shelf execution have become critical differentiators. Retailers are under constant pressure to ensure planogram compliance, the agreement between the planned shelf layout and the actual physical arrangement, to maximize sales and customer satisfaction. However, maintaining this compliance is operationally difficult; industry data indicates that shelf layouts can drift out of compliance by as much as 10% each week without consistent monitoring (Sumanth, 2025).

Conventionally, stores rely on manual auditing to identify issues such as Out-of-Stock (OOS) items or misplaced products. This process is inherently flawed due to human limitations; recent reports highlight that manual processes often miss up to 18% of non-compliance instances, reducing the accuracy of store-level insights (Sumanth, 2025). Furthermore, the lack of real-time visibility for headquarters and the high labor cost associated with frequent manual checks make this approach unsustainable for large retail networks.

The necessity for automation, driven by these operational challenges, has catalyzed significant investment in Artificial Intelligence (AI) solutions for the retail sector. This massive market covering applications from personalization to in-store monitoring is projected to reach approximately USD 127.2 billion by 2033 (Market.us, 2024). Within this broader AI transformation, solutions for physical retail monitoring rely heavily on Computer Vision (CV), making it a critical growth segment. Consequently, the specific market for automated shelf monitoring solutions is projected to surge from USD 2.05 billion in 2025 to USD 12.56 billion by 2033, where the segment of Object Detection & Tracking technology continues to dominate, having accounted for the largest revenue share of 29.5% in 2024 due to its fundamental role in retail automation (Grand View Research, 2025).

However, the transition to automation faces a specific, persistent computational challenge: fine-grained product recognition. Modern grocery stores contain thousands of Stock Keeping Units (SKUs), many of which exhibit high intra-family and inter-class similarity. A

particularly difficult scenario arises with product variants from the same brand that differ only in weight or grammage but share identical packaging graphics (Srivastava & Kumar, 2021). Recent computer vision studies emphasize that distinguishing these visually similar products is a "fine-grained task" because products from the same manufacturer or category can appear similar based on their shapes, colors, and sizes, making them difficult to distinguish solely on their visual representations (Budimir et al., 2025). In these challenging conditions, even current state-of-the-art detectors can confuse one grammage variant with another, leading to incorrect inventory data.

This inherent difficulty is compounded by two major practical limitations in applied deep learning. First, the success of deep learning models critically depends on vast and varied training data, yet collecting such data for every product variant under all real-world conditions is often impractical, leading to the challenge of data scarcity. Second, optimal model performance requires finding the perfect configuration for the learning process, necessitating a systematic optimization strategy to prevent overfitting and ensure the model learns the subtle visual cues between fine-grained classes effectively (Saputra, et al., 2025). The core failure point in current retail automation is not the absence of detection, but the low confidence and inconsistency when distinguishing between visually similar product variants due to these data and optimization shortcomings.

This research arises from these operational and technical challenges. To address these issues, this research employs an approach that emphasizes fine-grained class definition and manual annotation at the dataset level, where detailed data preprocessing, optimized model training, and controlled post-processing are used to build a detector capable of distinguishing visually similar product variants. By focusing specifically on distinguishing product variants with similar packaging designs but different weight categories, the study aims to address one of the most persistent issues in retail shelf automation. Improving the accuracy of fine-grained product detection promises significant benefits: more reliable inventory data, reduced labor requirements, improved store consistency, and better decision-making for supply chain operations. Ultimately, the motivation behind this research aligns with the broader goal of enabling smarter, more efficient retail environments capable of maintaining accuracy at scale.

1.2 Problem Statement

Based on the background outlined in the previous section, the core issue identified is the inability of current detection systems to reliably distinguish retail products that share identical

visual features but differ in specifications. The specific problems to be addressed in this research are formulated as follows:

a. Standard object detection architectures struggle to accurately classify fine-grained product variants that share nearly identical packaging designs, as the subtle visual cues distinguishing them are often lost during feature extraction, leading to high inter-class confusion.
b. The effectiveness of deep learning models in the retail domain is severely hindered by data scarcity and class imbalance, where collecting large-scale, annotated datasets for every specific product variant is impractical, resulting in poor model generalization on minority classes.
c. The lack of a systematic optimization strategy for hyperparameters and training configurations often causes detection models to converge sub-optimally, resulting in inconsistent prediction confidence and localization errors when facing visually ambiguous products.

Based on these problem statements, the primary question seeks to be answered is: "How can automated fine-grained retail detection accurately distinguish visually similar product variants while also overcoming the potential performance limitations caused by data scarcity?

1.3 Research Objective

The primary objective of this research is to develop a robust detection framework capable of distinguishing fine-grained retail product variants with high accuracy. Specifically, the research aims to achieve the following:

a. To develop and evaluate a fine-grained detection pipeline by implementing YOLOv12 and Faster R-CNN, analyzing their distinct feature extraction capabilities to establish a robust baseline for discerning subtle packaging differences between weight variants.
b. To optimize model generalization under data-constrained conditions by applying Context-Aware Copy-Paste augmentation, aiming to synthesize realistic training samples.
c. To implement a multi-stage systematic optimization strategy using the Tree-Structured Parzen Estimator (TPE) to tune both model hyperparameters and augmentation configurations, aiming to fully exploit the detection potential of each architecture beyond its standard baseline configuration.

d. To enhance detection consistency and reduce false positives by applying Weighted Boxes Fusion (WBF) as an ensemble post-processing strategy, validating the benefit of leveraging complementary architectural strengths to achieve a more robust prediction.

To provide a clearer overview of the research aim, this study considers a retail shelf image containing multiple visually similar product variants as the primary input. The expected output of the proposed framework is a set of accurately localized bounding boxes, each correctly classified according to its specific product variant and grammage. This input–output formulation illustrates the core objective of the research, namely enabling automated retail systems to reliably distinguish fine-grained product variants under visually ambiguous and data-constrained shelf conditions.

1.4 Research Contribution

The outcomes of this research provide several contributions to the field of computer vision and retail automation, specifically:

a. Methodological Insight: Demonstrating the efficacy of systematic hyperparameter tuning, data augmentation, and post-processing strategy as a viable solution for overcoming data scarcity and improving generalization in retail environments.
b. Reliability Improvement for Retail Automation: Offering a strategy to minimize confusion among visually similar products, serving as a reference for scalable automated shelf monitoring systems.

1.5 General Research Methodology

To achieve the objectives outlined previously, this research follows a systematic and structured workflow. The methodology encompasses the entire research lifecycle, from initial problem identification to the final synthesis of results, detailed as follows:

a. Data Collection: The dataset used in this research consists of retail shelf images containing products with a specific focus on product variants that share highly similar packaging designs but differ in weight or grammage. The data are collected to represent real-world shelf conditions, where multiple product variants from the same brand appear simultaneously within dense shelf layouts.
b. Data Preprocessing: Data preprocessing includes data refinement, data labeling, and data splitting. This stage ensures that each product instance is accurately labeled according to its specific product variant and grammage.

c. Model Training: The model development and training were executed in a High-Performance Computing (HPC) environment to handle the computational demands of systematic hyperparameter tuning and complex architecture training. The training infrastructure utilizes an Ubuntu 20.04 LTS server equipped with NVIDIA Tesla V100-SXM2 GPUs (32GB VRAM), powered by an Intel Xeon Gold 6138 CPU and 251 GB of System RAM. The software stack relies on Python 3.10 within isolated environments utilizing Ultralytics and TorchVision frameworks.
d. Post-Processing: To improve detection consistency and reduce prediction ambiguity, an ensemble post-processing strategy is applied using Weighted Boxes Fusion (WBF). This method combines the inference outputs of the optimized detection models by leveraging their confidence scores and spatial agreement, resulting in more reliable final predictions.
e. Performance Evaluation: The performance of the proposed detection pipeline is evaluated using confusion matrix analysis, with particular emphasis on False Negatives (FN) and False Negative Rates (FNR) to assess missed detections and fine-grained misclassification. Standard detection metrics, including mean Average Precision (mAP), are also reported to provide a comprehensive assessment of model performance across baseline, optimized, and ensemble configurations.

1.6 Report Structure

This essay is organized into five chapters, structured systematically from the theoretical foundation to the experimental findings and conclusions. The outlines are as follows:

a. Chapter I: Introduction, this chapter serves as the foundation of the research. It outlines the background of the study, emphasizing the operational and technical challenges of the modern retail environment. It defines the problem statements, research questions, scope of work, and the specific objectives and contributions aimed at solving the issues of fine-grained retail detection.
b. Chapter II: Literature Review, this chapter provides the theoretical basis for the study. It reviews object detection architectures, conducting a comparative analysis of distinct detection paradigms to evaluate their trade-offs. Furthermore, we explore methods to enhance model performance, including systematic hyperparameter optimization techniques, dataset-level improvement, and post-processing strategies. The chapter concludes by providing the scientific justification for the specific algorithms and methods selected for implementation in this research.

c. Chapter III: Research Methodology, this chapter details the technical implementation and experimental framework. It explains the end-to-end pipeline, starting from data acquisition (private and field collection) and preprocessing (manual annotation and refinement). The core discussion focuses on the multi-stage training strategy, including the implementation of Optuna for hyperparameter and augmentation tuning, the application of dataset-level Copy-Paste augmentation, and the Weighted Boxes Fusion (WBF) ensemble strategy.
d. Chapter IV: Results and Discussions, this chapter presents the experimental findings and in-depth analysis of the study. Utilizing Confusion Matrix to evaluate False Negatives (FN) and False Negative Rates (FNR) to identify missed detections and misclassification patterns. Subsequently, it reports the quantitative performance of the models using standard metrics such as mean Average Precision (mAP). The discussion highlights the effectiveness of the optimization strategies in reducing the rate of overlooked product variants compared to baseline configurations.
e. Chapter V: Conclusion, this chapter summarizes the key findings in relation to the initial research objectives. It draws conclusions on the efficacy of the proposed optimization pipeline and outlines the limitations encountered during the study. Finally, it offers recommendations for future research directions to further enhance the scalability and accuracy of retail shelf monitoring systems.

CHAPTER II
LITERATURE REVIEW

2.1 The Problem and Challenges in Fine-Grained Retail Product Detection

Although the operational challenges of retail shelf monitoring have been widely recognized, understanding why these problems persist requires examining the issue from a technical and computational perspective. Beyond labor intensity and human error, the core difficulty lies in the visual characteristics of retail products themselves. Fine-grained variants often share nearly identical packaging designs, making them difficult for both humans and automated systems to discriminate. This section contextualizes the technical foundations of the problem and outlines the scientific challenges that motivate the need for a more robust detection approach.

In practice, many retail stores still rely heavily on manual shelf monitoring processes. Store staff are tasked with checking product availability, verifying price and placement accuracy, and identifying misplaced or missing items. While this approach has been used for decades, it suffers from fundamental limitations: it is slow, labor-intensive, prone to human error, and difficult to scale across large store networks (Ailet, 2025). As the number of retail products continues to grow every year, the burden placed on workers becomes increasingly unrealistic.

This challenge becomes even more significant in product categories such as beverages, snacks, and others, where manufacturers frequently release multiple grammage variants (e.g., 1 kg, 750 g, 350 g, 120 g) with minimal visual differences, these fine-grained variations usually arise from slight variations in text, size, or color of the package (Peng et al., 2021). Subtle distinctions in text size, color shade, or small numerical indicators may be the only features separating one variant from another. This creates a fundamental challenge for both automated systems and human workers, as the retail environment frequently houses visually similar products. As a result, misidentification is common, both during manual inspection and even when using traditional computer vision methods that are not optimized to distinguish between products with such subtle visual differences (Pettersson et al., 2024).

Figure 2.1 Example of a similarly designed product with size variants

Source: (Srivastava & Kumar, 2021)

In wholesale settings, where products are arranged in larger quantities and shelves span greater lengths, these issues are amplified. Detecting and differentiating visually similar product variants becomes even more critical, as inaccuracies at this scale can lead to downstream errors in distribution planning and warehouse stocking.

As retail environments grow more complex, traditional monitoring approaches become increasingly unsustainable. Industry trends point toward automation as a strategic necessity rather than a technological luxury. Automated product detection systems enable retailers to perform frequent, objective, and scalable shelf monitoring without relying solely on human labor. However, developing such systems is far from trivial. Fine-grained retail detection requires models capable of identifying subtle visual cues within highly uniform product families, a level of precision not typically required in conventional object detection tasks.

Despite the urgency, research specifically focusing fine-grained weight variant detection remains limited. A prior study introduced a two-stage approach that incorporated an additional post-detection “reasoning” module after the initial detection stage (Srivastava & Kumar, 2021). The integration of such specialized modules underscores the complexity of this task. Looking at the broader landscape of retail automation, it is evident that several

fundamental hurdles ranging from the extreme visual similarity between variants to the practical constraints of data acquisition and the high resource demands required for iterative model optimization still pose significant barriers to reliable deployment.

To address these challenges, an effective solution for fine-grained retail product detection must satisfy the following key criteria derived from the characteristics of modern retail environments:

a. The detection model must achieve high accuracy in distinguishing fine-grained grammage variants, as retail products often differ only through subtle visual cues such as small numeric text or minor color shifts, factors that have been shown to significantly increase misclassification rates in fine-grained retail datasets.
b. The detection pipeline should be designed to perform effectively under limited data availability, which is a common constraint in certain object detection scenarios. In practical retail settings, collecting large-scale and diverse datasets for each grammage variant is challenging due to annotation costs and the structural limitations of in-store data acquisition. Product arrangements within a single store typically follow a fixed planogram, resulting in highly similar visual layouts across captured images. Consequently, acquiring a large number of images from the same store often leads to redundant samples with limited visual variation, while obtaining data across multiple stores with differing planograms requires substantial effort and coordination. Consequently, the solution should emphasize the importance of optimization as a means to extract maximal performance from constrained datasets, rather than relying solely on data scale. Addressing this challenge is essential to ensure that the proposed approach remains realistic, reproducible, and applicable to real-world retail environments where data resources are inherently limited.
c. The system should maintain a degree of computational efficiency that allows the model to be trained and iterated effectively within the constraints of the environment. In practice, model development often involves repeated experimentation, tuning, and refinement, making it essential that the computational workflow remains manageable and does not impose excessive overhead. Maintaining this balance enables a more robust development cycle and supports the use of well-established architectures that are known for their stable performance and mature implementation support. As a result, improved computational efficiency can facilitate more frequent optimization cycles, which may indirectly contribute to better model accuracy and overall detection performance.

2.2 Object Detection Architectures

Object detection is a fundamental task in computer vision that focuses on identifying the presence, category, and spatial location of objects within an image. Unlike image classification, which assigns a single label to an entire image, object detection simultaneously performs object localization and object classification for multiple objects in a scene (Zhao et al., 2019). This dual objective makes object detection a core component of high-level image understanding and a critical building block for many practical applications, including image analysis, video understanding, and automated visual monitoring systems.

Early object detection approaches relied primarily on handcrafted features and shallow learning models, which limited their ability to handle complex visual variations such as changes in scale, illumination, background clutter, and object occlusion. The introduction of deep learning, particularly convolutional neural networks (CNNs), marked a significant shift by enabling models to automatically learn hierarchical and high-level feature representations directly from data. As summarized by Zhao et al. (2019), deep learning–based object detection frameworks provide a more flexible and powerful foundation for visual recognition tasks, allowing detection systems to achieve higher robustness and accuracy in real-world scenarios.

2.2.1 Object Detection Paradigms

Object detection architectures have evolved through several distinct paradigms, each reflecting different design priorities in terms of detection formulation, feature representation, and computational structure. Broadly, modern object detection methods can be categorized into three major architectural families: two-stage detectors, single-stage detectors, and transformer-based detectors. Understanding these paradigms provides an architectural foundation for analyzing model selection in fine-grained retail product detection tasks, where visual similarity and scale variation present additional challenges.

Two-stage detectors represent one of the earliest structured approaches to deep learning–based object detection. This paradigm decomposes the detection task into two sequential processes: region proposal generation followed by region-wise classification and localization. The foundational R-CNN framework introduced this concept by applying convolutional neural networks to candidate object regions, significantly advancing detection accuracy compared to hand-crafted pipelines (Girshick et al., 2014). Subsequent refinements improved efficiency through shared feature computation and end-to-end training, most notably in Fast R-CNN (Girshick, 2015). The integration of a Region Proposal Network (RPN) into the detection

pipeline marked a major architectural milestone in Faster R-CNN, enabling proposal generation and classification to be jointly optimized within a unified framework (Ren, He, Girshick, & Sun, 2016). This architectural lineage emphasizes structured region reasoning and precise localization through explicit proposal mechanisms.

In contrast, single-stage detectors reformulate object detection as a unified prediction problem, removing the explicit region proposal stage entirely. Instead of generating candidate regions, these models directly predict object classes and bounding box coordinates across dense spatial locations in a single forward pass. Early representatives such as the Single Shot MultiBox Detector (SSD) introduced multi-scale feature maps to support detection across varying object sizes (Liu et al., 2016). The YOLO (You Only Look Once) family further advanced this paradigm by framing detection as a global regression task over a grid-based representation, allowing simultaneous prediction of object locations and categories within a single network evaluation (Redmon et al., 2016). Over successive versions, the YOLO architecture has undergone continuous refinements in feature extraction strategies, multi-scale representation, and training stability, reflecting ongoing efforts to balance detection robustness and computational efficiency. Another notable single-stage approach, RetinaNet, introduced focal loss to address class imbalance in dense prediction settings, highlighting how loss design can play a critical role within this architectural paradigm (Lin et al., 2018).

More recently, transformer-based detectors have introduced a fundamentally different architectural perspective by leveraging attention mechanisms to model global image context. Unlike CNN-based detectors that rely on anchor boxes or heuristic post-processing steps, transformer-based methods adopt set-based prediction formulations that directly output object hypotheses. The Detection Transformer (DETR) framework pioneered this approach by combining convolutional feature extraction with a transformer encoder–decoder architecture, enabling end-to-end object detection without non-maximum suppression (Carion et al., 2020). Subsequent works addressed challenges related to training efficiency and convergence through architectural and optimization improvements, such as sparse attention mechanisms in Deformable DETR and enhanced matching and denoising strategies in later variants (Zhu et al., 2021). Extensions such as RT-DETR further adapt transformer-based designs toward reduced computational overhead, reflecting continued exploration of attention-driven detection under practical constraints (Zhao et al., 2024).

To consolidate the architectural review presented in the previous sections, this subsection provides a structured comparison of key object detection models across the three major

paradigms: two-stage, single-stage, and transformer-based detectors. While each architecture embodies a distinct design philosophy, their practical suitability for fine-grained retail product detection varies significantly based on accuracy requirements, hardware constraints, and deployment scenarios.

Table 2.1 summarizes the strengths and weaknesses of representative models, Faster R-CNN, SSD, YOLO, RetinaNet, DETR, and DINO, highlighting their relative advantages and limitations in contexts relevant to retail shelf analysis.

Table 2.1 Comparison of object detection architectures

Architecture / Model	Strengths	Weaknesses
Faster R-CNN	+ High detection accuracy due to proposal-based region reasoning (Ren et al., 2016).	– Slow inference compared to single-stage detectors (Huang et al., 2017).
SSD (Single Shot MultiBox Detector)	+ Performs object localization and classification in a single forward pass, enabling fast inference suitable for real-time applications (Liu et al., 2016).	– Shows clear weakness in detecting small objects, often missing small instances in cluttered scenes (Huang et al., 2017). – Overall lower accuracy compared to two-stage detectors (Huang et al., 2017).
YOLO (You Only Look Once)	+ Extremely fast detection due to regression-based formulation (Redmon et al., 2016). + Single-stage directly looks at the entire image which enables real-time inference (Redmon et al., 2016). + Strong detection accuracy on benchmark datasets (Bochkovskiy et al., 2025)	– Struggles to localize objects accurately especially small objects (Redmon et al., 2016).
RetinaNet	+ Focal Loss handles heavy class imbalance effectively (Lin et al., 2018). + Achieves detection accuracy comparable to two-stage detectors while maintaining a single-stage architecture (Lin et al., 2018).	– Higher computational cost due to dense anchor evaluation, which may limit real-time deployment (Lin et al., 2018).
DETR	+ End-to-end object detection without hand-crafted components such as anchors and NMS (Carion et al., 2020).	– Extremely slow convergence (Zhu et al., 2021). – High computational cost due to long-training schedule (Carion et al., 2020).

	+ Strong global context modeling through self-attention, enabling global image reasoning (Carion et al., 2020).02/09/2026 07:31:05	

Based on the comparative analysis presented in Table 2.1, this study focuses on YOLO and Faster R-CNN as the primary object detection models with the objective of achieving the highest possible detection accuracy in fine-grained retail product recognition. The selection is driven primarily by accuracy-related considerations rather than inference speed, as the task involves distinguishing visually similar product variants with subtle differences in size, text, and packaging layout.

Faster R-CNN is included due to its consistently strong detection accuracy and robust proposal-based region reasoning. Its two-stage formulation enables more precise localization and classification by explicitly focusing on candidate regions, which is particularly advantageous in dense retail scenes containing overlapping products and fine-grained visual cues. As a result, Faster R-CNN serves as a reliable accuracy-oriented baseline that has been extensively validated across a wide range of detection benchmarks.

YOLO is selected as a complementary single-stage detector that has demonstrated competitive detection accuracy in modern variants, approaching that of two-stage architectures while retaining a unified detection formulation. Recent YOLO generations show substantial improvements in feature representation, multi-scale reasoning, and training stability, enabling strong performance on challenging benchmark datasets. This makes YOLO a suitable candidate for evaluating whether a single-stage architecture can achieve high accuracy in fine-grained retail detection scenarios.

Other architectures, including SSD, RetinaNet, and transformer-based detectors (DETR), are not selected as primary models despite their notable strengths. SSD exhibits clear limitations in detecting small and fine-grained objects, which are common in retail product imagery. RetinaNet, while achieving accuracy comparable to two-stage detectors, introduces additional computational complexity due to dense anchor evaluation without offering a clear accuracy advantage over Faster R-CNN in fine-grained settings. Transformer-based detectors such as DETR represent a more recent architectural paradigm and, although conceptually elegant, require long training schedules and substantial computational resources. Moreover, their relative architectural novelty means that their implementations and training behaviors are

not yet as mature or extensively validated as those of CNN-based detectors in fine-grained retail contexts.

Therefore, the combination of Faster R-CNN and YOLO provides a balanced experimental framework that prioritizes detection accuracy, architectural maturity, and empirical reliability, aligning with the primary objective of this study.

2.2.2 Theoretical Basis of Selected Object Detection Models

A. Faster R-CNN

Faster R-CNN is one of the most established two-stage object detection frameworks and serves as a strong baseline for high-accuracy detection tasks. As discussed in Section 2.2.1, Faster R-CNN extends R-CNN and Fast R-CNN by integrating a Region Proposal Network (RPN) into a unified, end-to-end architecture (Ren et al., 2016).

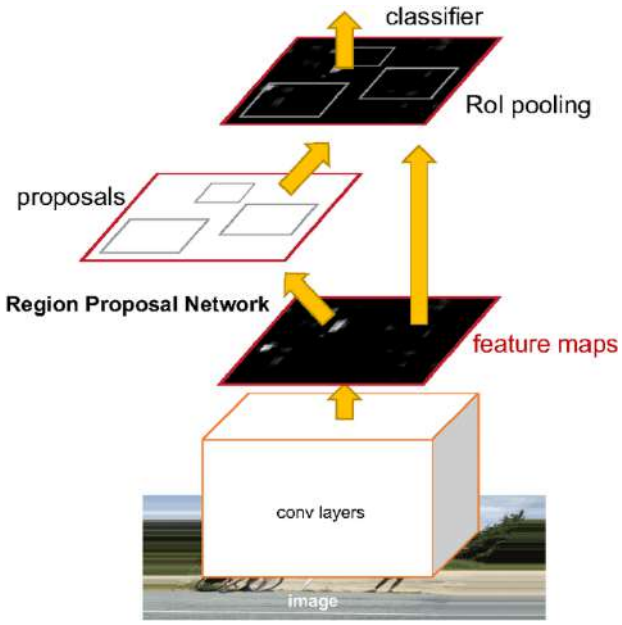


Figure 2.2 The architecture of Faster R-CNN

Source: (Ren et al., 2016)

The architecture consists of four major components. First, a convolutional backbone extracts hierarchical feature maps from the input image. Commonly used backbones include ResNet-50, ResNet-101, and ResNeXt-101 (often combined with a Feature Pyramid Network),

which enhance detection of objects across multiple scales. Second, the Region Proposal Network (RPN) slides a small convolutional window over the feature map to predict objectness scores and bounding-box offsets for a set of anchors of varying scales and aspect ratios. Third, the top proposals are processed by the RoI Align module, which corrects the misalignment introduced by quantization in earlier RoI Pooling. Finally, the detection head performs classification and bounding-box regression using the aligned features, producing the final refined predictions.

In this study, Faster R-CNN is implemented using the TorchVision framework, utilizing a ResNet-50 backbone and an enhanced Feature Pyramid Network (FPN). Faster R-CNN is particularly well suited for fine-grained retail product detection because its two-stage formulation enables more precise modeling of localized visual details. The Region Proposal Network (RPN) identifies candidate regions, allowing the detector to focus on subtle cues such as numeric weight text, color variations, or small layout differences that distinguish one grammage variant from another. Although its computational cost is higher than that of single-stage detectors, Faster R-CNN offers the accuracy-oriented characteristics.

B. YOLOv12

YOLOv12 is a modern single-stage object detection framework designed to combine the efficiency of convolution-based YOLO models with the representational benefits of attention mechanisms. Unlike earlier YOLO variants that rely purely on convolutional operations, YOLOv12 introduces an attention-centric design aimed at improving feature modeling while maintaining real-time inference characteristics (Tian et al., 2025).

The architecture of YOLOv12 consists of three main components. First, the backbone incorporates the Residual Efficient Layer Aggregation Network (R-ELAN), an improved version of ELAN designed to address instability issues that arise when deep aggregation structures interact with attention modules. R-ELAN introduces a residual shortcut with a scaling factor, allowing gradients to flow more consistently throughout the block and stabilizing training in deeper networks. In addition, the aggregation strategy is redesigned into a bottleneck-style structure: instead of processing multiple split paths, R-ELAN applies a transition layer to form a single intermediate feature map, processes it through subsequent convolutional blocks, and concatenates the outputs afterward. This modification preserves the feature integration capability of ELAN while reducing computational cost, parameters, and

memory usage. As a result, R-ELAN provides a more stable and efficient backbone foundation for attention-centric detection in YOLOv12 (Tian et al., 2025).

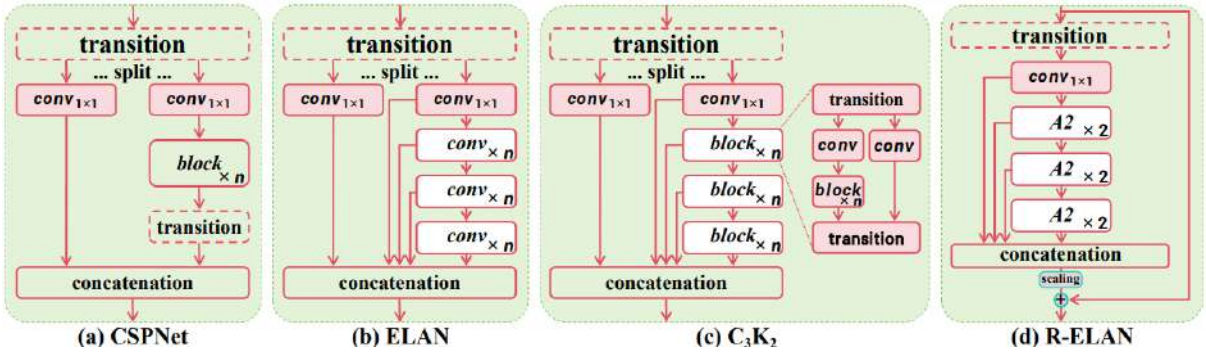


Figure 2.3 Architectural comparison of efficient layer aggregation modules

Source: (Tian et al., 2025)

Second, the network integrates Area Attention (A2), a simplified local attention mechanism that expands the receptive field by partitioning feature maps into spatial regions. This allows YOLOv12 to encode broader contextual cues more efficiently than traditional self-attention, which is typically too computationally expensive for real-time detectors. Third, the decoupled detection head separates classification and regression branches, improving convergence stability and enabling more precise bounding-box localization. Additional optimizations, such as FlashAttention-based acceleration, lightweight positional encoding through large-kernel convolutions, and adjustments to the MLP rasio, contribute to further efficiency gains. These enhancements allow YOLOv12 to outperform previous YOLO generations in both accuracy and latency across multiple model scales, making it a suitable single-stage detector for experimentation under limited computational resources.

2.3 Hyperparameter Optimization

Hyperparameter optimization (HPO) is a critical process in deep learning that aims to automatically determine optimal configurations for parameters that cannot be learned directly during training. These parameters include both structure-related settings, such as network depth and activation functions, and training-related parameters, such as learning rate, batch size, and optimizer selection (Yu & Zhu, 2020). Improper hyperparameter selection can significantly degrade model performance, making the training process inefficient, unstable, and difficult to reproduce.

Yu and Zhu (2020) emphasize that designing and training deep neural networks remains a challenging and unpredictable process, often relying heavily on empirical knowledge and trial-and-error. Manual hyperparameter tuning requires substantial expertise and computational resources, yet it often yields suboptimal configurations rather than truly optimal solutions. As a result, automated hyperparameter optimization has emerged as a crucial component of modern deep learning workflows, aiming to reduce human intervention, improve training efficiency, and enhance the reproducibility and reliability of experimental results.

2.3.1 Hyperparameter Optimization Methods

Hyperparameter optimization (HPO) techniques can be broadly categorized into stochastic search, Bayesian optimization, and multi-fidelity optimization, each aiming to systematically explore hyperparameter spaces to improve model performance while controlling computational cost. In deep learning workloads, such as fine-grained retail product detection, the choice of hyperparameter optimization method is critical due to the high dimensionality of the search space and the substantial resources required for training object detection models.

Early stochastic search methods rely on simple and direct sampling strategies without incorporating probabilistic modeling of the objective function. Grid search represents the most straightforward approach, in which all predefined combinations of hyperparameters are exhaustively evaluated. While this guarantees complete coverage of the search space, it becomes computationally impractical as the number of hyperparameters increases, because it allocates equal effort to all configurations regardless of their potential performance (Bergstra & Bengio, 2012). Random search improves upon this by sampling configurations randomly, enabling broader exploration of high-dimensional spaces and often finding effective configurations with fewer evaluations (Bergstra & Bengio, 2012).

Bayesian optimization (BO) provides a more principled and sample-efficient approach to hyperparameter tuning by constructing a probabilistic surrogate model of the objective function and using it to guide the search toward promising regions of the hyperparameter space. Classical Bayesian optimization often employs Gaussian Processes (GP) as surrogate models, which estimate both the expected performance and uncertainty for candidate configurations. The acquisition function then balances exploration and exploitation to select the next hyperparameter set for evaluation (Snoek et al., 2012).To better handle high-dimensional, hierarchical, and conditional hyperparameter spaces, the Tree-structured Parzen Estimator (TPE) was introduced. TPE models two conditional probability densities: $l(x)$ for

configurations associated with good performance and $g(x)$ for those associated with poor performance. New configurations are sampled to maximize the ratio $l(x)/g(x)$, allowing the search to focus on regions that are statistically likely to yield improvements (Bergstra & Bengio, 2012). The Optuna framework implements TPE as its primary sampling algorithm, incorporating practical features such as pruning of unpromising trials, dynamic search spaces, and efficient logging to optimize computational resources (Akiba et al., 2019).

Multi-fidelity optimization methods offer an efficient alternative by evaluating candidate hyperparameter configurations using reduced computational budgets before committing to full training. Hyperband, for example, introduces an early-stopping mechanism inspired by multi-armed bandit strategies. It initially evaluates a large pool of configurations at low cost, typically through fewer epochs or smaller subsets of data, eliminating poorly performing configurations early while allocating additional resources to promising ones (Li et al., 2018).Building on this concept, BOHB—short for Bayesian Optimization with Hyperband, integrates TPE-based Bayesian sampling with Hyperband's multi-fidelity scheduling. This combination allows the search process to leverage prior knowledge from previous trials to guide sampling toward high-potential configurations while maintaining computational efficiency through early stopping, resulting in scalable and robust hyperparameter optimization for complex deep learning models (Falkner et al., 2018).

The diversity of hyperparameter optimization techniques reflects the wide range of computational constraints, model complexities, and performance requirements encountered in modern deep learning workflows. Each method, whether stochastic, Bayesian, or multi-fidelity, offers a distinct balance between efficiency, adaptability, and computational overhead. For object detection tasks that involve long training times and high-dimensional search spaces, understanding these trade-offs is essential for selecting an appropriate optimization strategy.

To synthesize the strengths and weaknesses of the methods that have been discussed, this subsection provides a consolidated comparison covering all major optimization families evaluated in this research. Table 2.2 summarizes the key characteristics, advantages, and limitations of each method.

Table 2.2 Comparison of hyperparameter optimization methods

Method	Strengths	Weaknesses
Grid Search	+ Simple, easy to implement (Bartz-Beielstein & Zaefferer, 2023).	− Computationally inefficient for high-dimensional spaces

		(Bartz-Beielstein & Zaefferer, 2023). – Does not adapt based on previous evaluations (Bartz-Beielstein & Zaefferer, 2023).
Random Search	+ More efficient than grid search in high-dimensional settings (Bartz-Beielstein & Zaefferer, 2023).	– Still non-adaptive; sampling remains unguided (Bartz-Beielstein & Zaefferer, 2023).
Bayesian Optimization (GP-based)	+ Very sample-efficient; focuses on promising regions (Snoek et al., 2012). + Balances exploration-exploitation (Snoek et al., 2012).	– GP computational cost (Snoek et al., 2012). – Sensitive to high-dimensions & noise.
TPE (Tree-structured Parzen Estimator)	+ Handles continuous, categorical, and conditional hyperparameters naturally (Bergstra et al., 2011). + Efficient sampling and pruning algorithms reduce computational cost (Akiba et al., 2019).	– Performance may be less optimal in very small datasets compared to Gaussian Process (Akiba et al., 2019). – Less theoretically grounded than GP-based methods (Snoek et al., 2012).
Hyperband	+ Very efficient via principled early stopping (Li et al., 2018). + Can provide orders of magnitude speedups in practice (Li et al., 2018).	– Not inherently model-guided (Li et al., 2018). – Sensitive to resource budget setting. – Risk of premature elimination.
BOHB (Bayesian Optimization + Hyperband)	+ Combines TPE's adaptive sampling with Hyperband's cost efficiency (Falkner et al., 2018). + Better sample efficiency than Hyperband alone (Falkner et al., 2018).	– More complex scheduling and integration.

Based on the comparison presented in Table 2.2, this study adopts the Tree-structured Parzen Estimator (TPE) as the primary hyperparameter optimization strategy. TPE is selected due to its ability to efficiently explore high-dimensional, heterogeneous, and conditional hyperparameter spaces, which are characteristic of modern object detection frameworks. Detection models such as Faster R-CNN and YOLO involve a combination of continuous, discrete, and categorical hyperparameters, ranging from learning rates and weight decay to optimizer configurations and architectural settings, making TPE particularly well suited for this optimization scenario.

Compared to stochastic methods such as grid search and random search, TPE provides an adaptive and performance-driven sampling mechanism, allowing the optimization process to focus on regions of the search space that are statistically associated with improved detection accuracy. This capability is especially important in fine-grained retail product detection, where minor variations in training configuration can significantly influence convergence behavior. Unlike grid search, which becomes computationally prohibitive as dimensionality increases, and random search, which remains unguided, TPE leverages information from previous trials to progressively refine the search toward promising configurations.

While Gaussian Process–based Bayesian optimization offers strong theoretical guarantees and sample efficiency, its scalability limitations in high-dimensional and noisy objective functions reduce its practicality for large-scale object detection training. Multi-fidelity approaches such as Hyperband and BOHB further improve computational efficiency by incorporating early stopping mechanisms; however, these methods introduce additional scheduling complexity and tighter coupling between optimization logic and training pipelines. Such complexity can increase implementation overhead and complicate reproducibility, particularly in object detection tasks with long training cycles and sensitive convergence dynamics.

In contrast, TPE offers a balanced trade-off between optimization effectiveness, implementation simplicity, and robustness, enabling adaptive hyperparameter exploration without aggressive resource scheduling or complex orchestration mechanisms. This makes TPE well aligned with the objective of this study, which prioritizes achieving high final detection accuracy through stable and systematic optimization while maintaining a manageable and reproducible experimental workflow.

2.3.2 Theoretical Basis of Selected Hyperparameter Optimization Method

A. Optuna: A Next-Generation Hyperparameter Optimization Framework

Optuna is a next-generation hyperparameter optimization framework built around the define-by-run principle, which allows search spaces to be constructed dynamically during execution rather than being specified in advance. This flexibility makes Optuna highly suitable for deep learning workflows that involve complex, conditional, or interdependent hyperparameters. As discussed in Section 2.3.1, Optuna implements the Tree-structured Parzen

Estimator (TPE), a Bayesian sampling algorithm that models high- and low-performance regions separately (Akiba et al., 2019).

TPE operates efficiently in high-dimensional, non-convex search spaces by avoiding the computational overhead associated with Gaussian Process–based surrogate models. Instead, it uses non-parametric density estimators that adapt as more trials are evaluated. This makes TPE particularly effective for tuning deep learning models whose performance is sensitive to learning rate schedules, weight decay, momentum, and augmentation intensity. Optuna further incorporates an asynchronous pruning mechanism based on Successive Halving (ASHA), enabling early termination of underperforming trials and reducing computational cost.

The system architecture of Optuna consists of multiple workers that execute trials asynchronously, each interacting with a shared storage backend responsible for tracking trial history, pruning decisions, and sampling behavior. This architecture enables Optuna to scale seamlessly from local single-machine experimentation to distributed optimization environments without requiring complex configuration. Figure 2.4 provides an overview of this system design, illustrating how workers execute objective functions independently while exchanging progress information through shared storage.

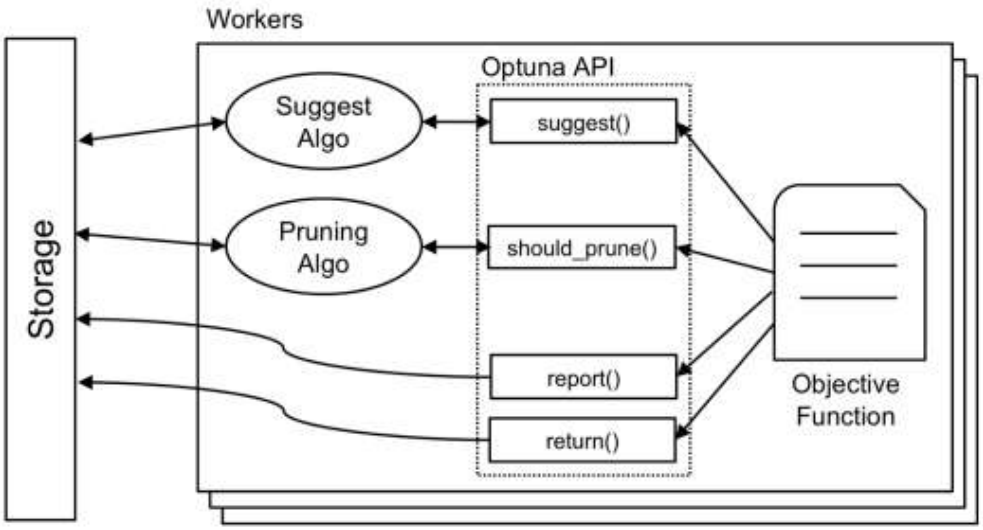


Figure 2.4 Overview of Optuna's system design

Source: (Akiba et al., 2019)

In this study, Optuna with the TPE sampler is used to optimize both model-level and augmentation-level hyperparameters. Its dynamic search-space construction, efficient

sampling behavior, and aggressive pruning strategy make it an effective and practical choice for iterative experimentation under computational constraints, while still offering reliable improvements in model performance.

2.4 Additional Methods to Improve Object Detection Performance

The performance of an object detection model is influenced not only by its architecture and training configuration, but also by the characteristics of the dataset used to train it. Limited samples, imbalanced class distributions, and insufficient visual diversity can constrain the model's ability to learn discriminative features. Dataset-level improvement techniques address these issues by increasing representativeness, variability, and coverage of training samples. Such techniques operate before the training stage and aim to strengthen the model's underlying feature learning without modifying the detector's architecture.

In addition to dataset-level improvements, modern object detection pipelines commonly employ post-processing mechanisms to refine the raw predictions produced during inference. Because detectors may output multiple bounding boxes for the same object or generate predictions with varying degrees of confidence, a consolidation step is required to produce coherent and interpretable results. Various post-processing approaches exist, including methods that adjust or merge predicted bounding boxes based on their spatial relationships or confidence distributions. These techniques serve as the final refinement stage and help ensure that the model's predictions are stable and aligned with the underlying visual content.

2.4.1 Dataset-Level Improvement: Data Augmentation

Dataset-level improvement refers to strategies that enhance the training data prior to model optimization, aiming to improve feature learning without modifying the detection architecture itself. In object detection tasks, particularly fine-grained retail product detection, the quality and diversity of training samples play a crucial role in determining the model's ability to distinguish visually similar object instances. Limitations such as restricted sample size, class imbalance, and insufficient appearance variation can lead to overfitting and weak generalization. Data augmentation addresses these issues by increasing variability and representativeness within the training dataset.

Data augmentation techniques can be broadly categorized based on the granularity at which transformations are applied. Pixel-level and image-level augmentations modify the visual appearance of entire images through operations such as color jittering, geometric

transformations, or noise injection. Beyond these traditional transformations, more advanced compositional augmentation strategies have been developed to synthesize new training samples by combining information from multiple images.

Mixing-based augmentation methods generate synthetic samples by blending two images and their corresponding labels. Mixup performs a linear interpolation between image pairs and label distributions, encouraging smoother decision boundaries and improved generalization. CutMix replaces a rectangular region of one image with a patch from another image while adjusting the labels proportionally based on the mixed area. These approaches have been widely adopted in deep learning pipelines to enhance robustness and reduce overfitting (Guo et al., 2023).

Another prominent category is grid-based compositional augmentation. Mosaic augmentation, introduced in the YOLOv4 training pipeline, combines four different images into a single composite arranged in a 2x2 grid. This process increases exposure to objects at different scales and spatial locations within a single training sample, enabling detectors to learn scale-invariant representations more effectively. Mosaic preserves instance boundaries while expanding contextual diversity across training iterations (Bochkovskiy et al., 2020).

Instance-level augmentation focuses on manipulating object instances rather than entire image regions. Copy-Paste augmentation extracts segmented object instances from source images and inserts them into target images while maintaining consistent annotations. By increasing object frequency and spatial diversity without altering bounding-box structure, this approach enhances the learning of object-centric features and has demonstrated effectiveness in object detection and instance segmentation benchmarks (Ghiasi et al., 2021).

Recent developments further extend instance-level augmentation by incorporating semantic and contextual awareness. Context-Aware Copy-Paste (CACP) integrates scene understanding mechanisms to guide object placement, ensuring that pasted instances align with the spatial and semantic structure of the target background. By selecting suitable background regions based on multimodal cues, CACP reduces visual inconsistencies and improves the realism of synthetic samples (Guo, 2025).

Adaptive augmentation strategies introduce feedback from model performance into the data generation process. The Multi-Stage Adaptive Copy-Paste (MSACP) framework applies copy-paste augmentation iteratively across training stages and dynamically adjusts the number of synthesized instances per class based on class-wise Average Precision (AP). Classes with lower

detection performance receive proportionally more augmented samples, enabling targeted correction of class imbalance during training (Yu et al., 2023).

Table 2.3 Comparison of data augmentation types

Augmentation	Strengths	Weaknesses
Mixup	+ Improves generalization by training on convex combinations of samples and labels (Zhang et al., 2018). + Encourages simple linear behavior between training examples, acting as a strong regularizer (Zhang et al., 2018). + Reduces memorization of corrupt labels and improves robustness to adversarial examples (Zhang et al., 2018).	– Produces unnatural and locally ambiguous samples that can confuse the model, especially for localization tasks (Yun et al., 2019). – Degrades object localization and object detection performance when used as ImageNet pretraining (Yun et al., 2019).
CutMix	+ Efficiently utilizes all pixels by replacing removed regions with patches from other images, avoiding information loss (Yun et al., 2019). + Improves object localization by forcing the model to recognize objects from partial views (Yun et al., 2019). + Consistently improves transfer learning performance for object detection when used as ImageNet pretraining (Yun et al., 2019). + Enhances robustness to adversarial attacks, occlusion, and out-of-distribution samples (Yun et al., 2019).	– Requires careful selection of mixing ratio; inappropriate variants or label strategies degrade performance (Yun et al., 2019).
Mosaic	+ Improves contextual and scale diversity by combining four images (Bochkovskiy et al., 2020). + Reduces reliance on large batch sizes via multi-image	– Produces synthetic layouts that deviate from natural scene distributions. – May introduce boundary artifacts due to grid-based composition.

	BN statistics (Bochkovskiy et al., 2020). + Empirically improves detection accuracy without increasing inference cost (Bochkovskiy et al., 2020).	
Copy-Paste	+ Increases data diversity by recomposing object instances into new backgrounds (Ghiasi et al., 2021). + Improves detection performance without modifying architecture (Ghiasi et al., 2021). + Mitigates class imbalance by adaptively increasing synthetic samples for low-performing classes, guided by class-wise performance metrics (X. Yu et al., 2023). + Context-aware variants improve semantic consistency between pasted objects and background scenes, reducing unrealistic object placement (Q. Guo, 2025).	– Random Copy-Paste may generate semantically inconsistent samples, which can negatively affect training quality. – Fixed Copy-Paste strategies ignore model training dynamics, leading to suboptimal augmentation at different training stages.

Based on the comparison presented in Table 2.3, instance-level Copy-Paste augmentation is selected as the primary dataset-level improvement strategy in this study. Mixing-based approaches such as Mixup and CutMix are effective regularization techniques that improve generalization by encouraging smoother decision boundaries and robustness. However, these methods operate by blending image content or replacing regions at the image level, which can distort local object structure and introduce ambiguity in spatial localization. Such effects are less desirable for object detection tasks that rely on precise bounding-box regression, particularly in fine-grained scenarios where small visual differences between object variants are critical.

Grid-based compositional augmentation, exemplified by Mosaic, improves scale diversity and contextual variation by combining multiple images into a single composite. While this strategy enhances robustness to object scale variation, its grid-based layout may introduce synthetic spatial arrangements that deviate from real-world retail shelf distributions. In fine-

grained retail product detection, where product placement follows relatively structured and repetitive patterns, excessive deviation from natural layouts may reduce the relevance of augmented samples.

In contrast, Copy-Paste augmentation operates at the instance level, explicitly recomposing object instances into new backgrounds while preserving bounding-box annotations and object integrity. This characteristic makes Copy-Paste particularly suitable for fine-grained retail detection, as it increases object frequency and spatial diversity without altering the visual structure of the products themselves. By maintaining object-centric consistency, Copy-Paste enables the model to focus on subtle visual cues, such as packaging text, color tone, and layout differences that are essential for distinguishing highly similar product variants. Moreover, adaptive variants of Copy-Paste allow augmentation to be directed toward low-performing classes, directly addressing class imbalance without uniformly perturbing the entire dataset.

To further enhance the effectiveness of instance-level augmentation, this study adopts the underlying principle of Context-Aware Copy-Paste (CACP). Although the full multimodal framework of CACP is not directly implemented, its core idea, maintaining semantic and contextual consistency between pasted objects and background scenes is particularly relevant in fine-grained retail environments. Retail shelves typically exhibit strong contextual regularities, such as brand-wise grouping and product family organization. Incorporating context-aware placement ensures that augmented samples remain visually plausible and aligned with real-world retail layouts, thereby improving the usefulness of synthetic data for learning discriminative features.

Overall, Copy-Paste augmentation, complemented by context-aware placement principles, provides a balanced trade-off between diversity enhancement and structural realism. This makes it more suitable than mixing-based or grid-based augmentation strategies for improving detection accuracy in fine-grained retail product recognition, where preserving object integrity and contextual coherence is essential.

2.4.2 Theoretical Basis of Selected Data Augmentation Method

A. Context-Aware Copy-Paste Data Augmentation

Copy-Paste data augmentation is an instance-level compositional technique designed to enrich the visual diversity of training data by transplanting object instances from one image onto another while preserving their original bounding-box annotations. Unlike pixel-level

mixing or region-level composition, Copy-Paste explicitly operates at the object instance level, allowing models to observe the same objects under new spatial arrangements, contextual backgrounds, and inter-object relationships.

The foundational formulation of this technique was introduced through the Simple Copy-Paste method (Bodla et al., 2017). In its basic pipeline, instance masks or bounding boxes are first extracted from a source image, after which the cropped object region is pasted onto a target background image. The corresponding bounding-box or segmentation annotations are then updated to reflect the composited instance. Although conceptually simple, this approach demonstrated notable improvements in object detection and instance segmentation benchmarks by increasing the frequency, spatial variety, and contextual diversity of underrepresented classes. The method requires no architectural modifications and can be applied independently of the base model, making it broadly compatible with various detection frameworks.

Subsequent research expanded upon this idea by addressing the semantic constraints of compositional augmentation. Context-Aware Copy-Paste (CACP) improves the realism and utility of synthesized images by leveraging multimodal scene–object compatibility models to select suitable target backgrounds and determine insertion locations (Ganaie et al., 2022). By aligning composited objects with environments that are semantically plausible, CACP mitigates visual artifacts that may arise from random placement. As a result, the generated images more closely reflect real-world scenes, providing higher-quality training samples for models that rely on subtle contextual cues.

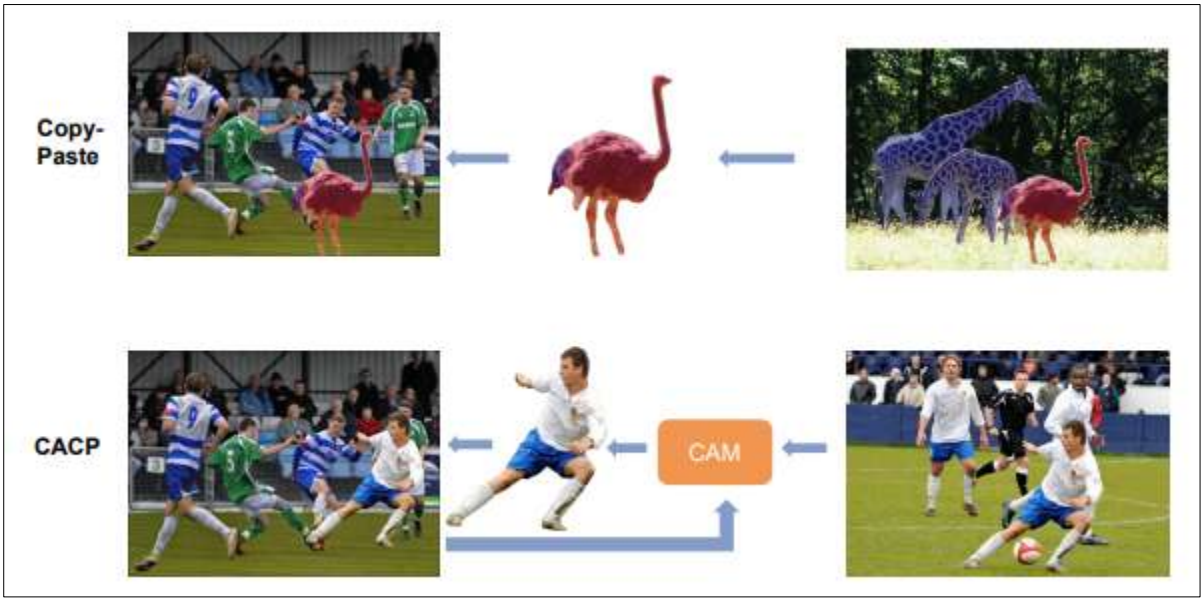


Figure 2.5 Comparison of Copy-Paste and Context-Aware Copy-Paste (CACP)

Source: (Guo, 2025)

2.4.3 Post-Processing-Level Improvement: Iou-Based Suppression and Ensemble Deep Learning Model

Post-processing-level improvement refers to techniques applied after an object detection model generates its raw predictions during inference. At this stage, detectors commonly produce multiple bounding boxes with associated confidence scores for the same object instance, often exhibiting significant spatial overlap. Without appropriate post-processing, these redundant predictions can lead to ambiguous outputs and degraded evaluation performance. Post-processing mechanisms therefore play a critical role in consolidating predictions into coherent and interpretable detection results.

One major category of post-processing techniques is IoU-based suppression. These methods resolve redundant detections by measuring the spatial overlap between predicted bounding boxes using the Intersection over Union (IoU) metric. The most widely adopted approach in this category is Non-Maximum Suppression (NMS). NMS iteratively selects the bounding box with the highest confidence score and removes all other boxes whose IoU with the selected box exceeds a predefined threshold. This procedure continues until no candidate boxes remain. While effective and computationally efficient, the hard suppression behavior of NMS can result in the removal of valid detections when objects are densely packed or when predicted boxes exhibit substantial overlap (Bodla et al., 2017).

To address this limitation, Soft-NMS was introduced as a modification of traditional NMS. Instead of discarding overlapping bounding boxes outright, Soft-NMS reduces their confidence scores as a continuous function of IoU with the highest-scoring box. The method employs linear or Gaussian decay functions to smoothly penalize overlapping detections rather than eliminating them. By avoiding abrupt suppression, Soft-NMS preserves potentially valid predictions and has been shown to improve detection performance across multiple benchmarks without requiring retraining of the underlying detector (Bodla et al., 2017).

Beyond suppression-based approaches, post-processing can also be performed through ensemble-based methods. Ensemble learning encompasses a broad family of techniques, commonly categorized into classical, general, and fusion-based methods (Ganaie et al., 2022). Classical and general ensemble strategies such as bagging or boosting operate at the model training level and are primarily designed for classification or regression tasks. Because these methods do not directly combine structured prediction outputs such as bounding boxes, they are not applicable as post-processing techniques in object detection.

In contrast, fusion-based ensemble methods operate directly on the prediction outputs of multiple detectors. These approaches aggregate bounding boxes, confidence scores, and spatial relationships across models to produce consolidated detection results. Fusion methods are specifically designed to handle bounding box geometry and overlapping predictions, making them suitable for object detection post-processing. By integrating predictions from multiple detectors, fusion-based ensembles aim to improve robustness and consistency by leveraging complementary information captured by different models.

Several fusion strategies have been proposed for bounding box aggregation. Non-Maximum Weighted (NMW) computes fused bounding box coordinates by weighting predictions according to confidence scores and IoU values, but does not iteratively update cluster centers or normalize contributions based on the number of models. Weighted Boxes Fusion (WBF) extends this idea by iteratively computing confidence-weighted averages of bounding box coordinates across all predictions within a cluster and incorporating normalization based on the number of contributing detectors. This formulation produces more stable and representative fused predictions compared to earlier fusion approaches (Solovyev, Wang, & Gabruseva, 2021).

Table 2.4 Comparison of post-processing methods

Augmentation	Strengths	Weaknesses
Non-Maximum Suppression (NMS)	+ Simple and widely used post-processing method for filtering redundant bounding boxes based on IoU and confidence ranking (Bodla et al., 2017). + Computationally efficient and suitable for standard single-model object detection pipelines (Solovyev et al., 2021).	– Relies on a hard IoU threshold, which is difficult to tune and may suppress valid detections in crowded or overlapping object scenarios (Bodla et al., 2017). – Discards overlapping predictions entirely, preventing aggregation of localization information from multiple models (Solovyev et al., 2021).
Soft-NMS	+ Mitigates the aggressive suppression behavior of NMS by decaying confidence scores instead of removing overlapping boxes (Bodla et al., 2017). + Improves detection accuracy on standard benchmarks without requiring retraining of the base detector (Bodla et al., 2017). + Can be applied as a drop-in replacement for NMS in existing detection pipelines (Bodla et al., 2017).	– Remains a greedy suppression method and does not produce averaged bounding boxes from multiple predictions.
Weighted Boxes Fusion (WBF)	+ Aggregates predictions from multiple detectors by computing confidence-weighted averaged bounding boxes rather than suppressing them (Solovyev et al., 2021). + Produces more accurate localization when combining outputs from multiple models or test-time augmentations (Solovyev et al., 2021). + Consistently outperforms NMS, Soft-NMS, and NMW	– Computationally slower than NMS and Soft-NMS, approximately three times slower on average. – Requires reasonably accurate base detectors; performance degrades with noisy predictions.

	in ensemble-based evaluations on COCO and Open Images benchmarks (Solovyev et al., 2021).	

Based on the comparison presented in Table 2.4, this study adopts Weighted Boxes Fusion (WBF) as the primary post-processing strategy for inference-stage refinement. Unlike IoU-based suppression methods such as NMS and Soft-NMS, which operate by removing or down-weighting overlapping predictions from a single detector, WBF is specifically designed to aggregate predictions originating from multiple models. This characteristic makes WBF particularly suitable for ensemble-based detection frameworks, where complementary strengths from heterogeneous architectures can be exploited (Solovyev et al., 2021).

In the context of this research, which combines YOLOv12 as a one-stage detector and Faster R-CNN as a two-stage detector, suppression-based methods are insufficient because they discard overlapping predictions rather than integrating them. Such behavior limits the ability to leverage diverse localization cues and confidence distributions produced by different detection paradigms. WBF addresses this limitation by computing confidence-weighted averages of bounding box coordinates across all contributing detectors, resulting in fused predictions that reflect consensus rather than dominance by a single model (Solovyev et al., 2021).

Furthermore, prior studies have shown that WBF consistently outperforms traditional suppression techniques, including NMS, Soft-NMS, and Non-Maximum Weighted (NMW) in ensemble settings across standard benchmarks such as COCO and Open Images (Solovyev et al., 2021). Although WBF incurs higher computational cost during inference, this trade-off is acceptable within the scope of this study, where inference efficiency is secondary to localization accuracy and robustness in fine-grained retail product detection. Consequently, WBF is selected as the post-processing method that best aligns with the ensemble-based design and accuracy-oriented objectives of this research.

2.4.4 Theoretical Basis of Selected Post-Processing Method

A. Weighted Boxes Fusion (WBF)

Weighted Boxes Fusion (WBF) is an ensemble-based post-processing algorithm designed to improve detection accuracy by merging predictions from multiple models rather than suppressing them. WBF aggregates bounding box coordinates and confidence scores across detectors that predict the same object (Solovyev et al., 2021).

WBF groups bounding boxes into clusters based on an IoU similarity threshold and computes a fused box for each cluster using a confidence-weighted average. The fused bounding box is defined as the confidence-weighted average of coordinates, as shown in Equation (2.1):

$$B_{fused} = \frac{\sum_{i=1}^{T} C_i \cdot B_i}{\sum_{i=1}^{T} C_i} \quad (2.1)$$

This equation ensures that bounding boxes with higher confidence contribute more strongly to the final fused coordinates. The fused box represents the "center of agreement" among all model predictions within a cluster.

The fused confidence score reflects the collective support from multiple detectors. It is computed as the average confidence of all boxes in the cluster, as shown in Equation (2.2):

$$C_{fused} = \frac{\sum_{i=1}^{T} C_i}{T} \quad (2.2)$$

Clusters supported by more detectors naturally produce higher fused confidence, while predictions from fewer models receive proportionally lower confidence.

The variables in Equation (2.1) and Equation (2.2) are defined as follows:

- B_{fused} represents the fused bounding box
- C_{fused} represents the final fused confidence value
- $B_i = (x_1, y_1, x_2, y_2)$ represents the coordinate tuple
- C_i represents the confidence score
- T represents the number of boxes grouped into the same cluster

Figure 2.6 illustrates the WBF algorithm, which iteratively groups bounding boxes, computes weighted coordinates, and produces a refined detection set without discarding any high-overlap predictions. Because WBF avoids the elimination of overlapping boxes, it is able to retain nuanced localization information that would otherwise be lost in suppression-based methods. This characteristic makes WBF particularly suitable for multi-model ensembles in dense scenes, such as retail shelves, where fine-grained product variants may appear closely aligned and require precise bounding box placement.

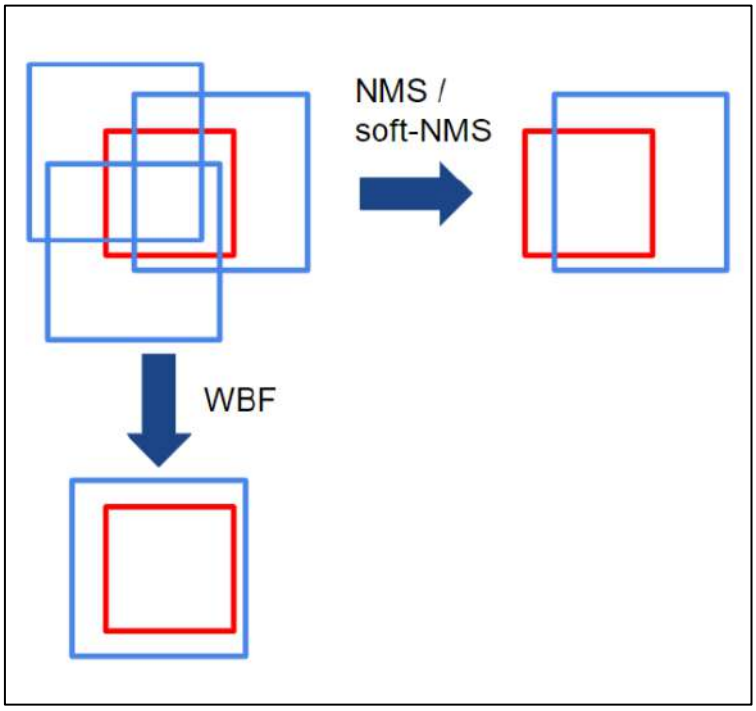


Figure 2.6 Comparison of NMS, Soft-NMS, and Weighted Boxes Fusion (WBF)
Source: (Solovyev et al., 2021)

Unlike suppression-based approaches, WBF does not require hyperparameter-sensitive IoU thresholds and is fully model-agnostic. It can be applied to any combination of detectors without requiring retraining or architectural modifications. The algorithm also maintains stable performance when combining models with differing bounding box shapes or localization tendencies, making it a robust and practical choice for fine-grained object detection tasks. As a result, WBF functions as a dedicated consensus-driven refinement stage in this study, serving as the final step in the detection pipeline before quantitative evaluation in the subsequent chapter.

2.5 Object Detection Performance Benchmark

Object detection requires well-defined performance benchmarks that quantify how effectively a model identifies and localizes objects. In this research, the evaluation begins with a diagnostic analysis using a Confusion Matrix. This tool is utilized to analyze class-specific errors and misclassification patterns, which are critical for understanding variant-level confusion in visually similar product families. By prioritizing this diagnostic step, the study explicitly addresses the fine-grained nature of the problem, identifying whether the model struggles with specific product variants.

Following this diagnostic assessment, the model's quantitative performance is evaluated using Average Precision (AP) and mean Average Precision (mAP). These metrics follow the conventions of modern detection benchmarks, such as Pascal VOC and COCO (Shah, 2022), and are widely recognized as the primary standards for measuring localization accuracy and prediction reliability. Unlike simple precision or recall metrics, AP and mAP provide a robust summary of detection quality across multiple thresholds, making them highly suitable for ensuring that fine-grained products are not only classified correctly but also localized with high precision. Together, these diagnostic and quantitative metrics form a comprehensive evaluation framework for assessing the baseline, optimized, and ensemble models developed in this study.

2.5.1 Diagnostic and Misclassification Metrics in Object Detection

To effectively analyze fine-grained errors, this research utilizes the Confusion Matrix as a visual diagnostic tool and the False Negative Rate (FNR) as a quantitative metric for misclassification.

A. Confusion Matrix

A confusion matrix is used in this research as a primary diagnostic tool to analyze class-level prediction behavior. It summarizes how often each object class is correctly detected or misclassified as another, providing insight into confusions between visually similar classes.

The matrix aligns predicted labels with ground-truth labels for all true-positive and false-positive detections. The orientation (rows and columns) depends on the implementation, but typically one axis represents ground-truth classes while the other represents predicted classes. Diagonal entries indicate correct predictions, while off-diagonal entries reveal misclassifications.

This diagnostic perspective is particularly important for fine-grained retail detection, where products may differ only by small numerical weight indicators or minimal packaging variations. Confusion matrix helps identify variant pairs that are frequently confused, guiding decisions on targeted augmentation, feature enhancement, or post-processing strategies.

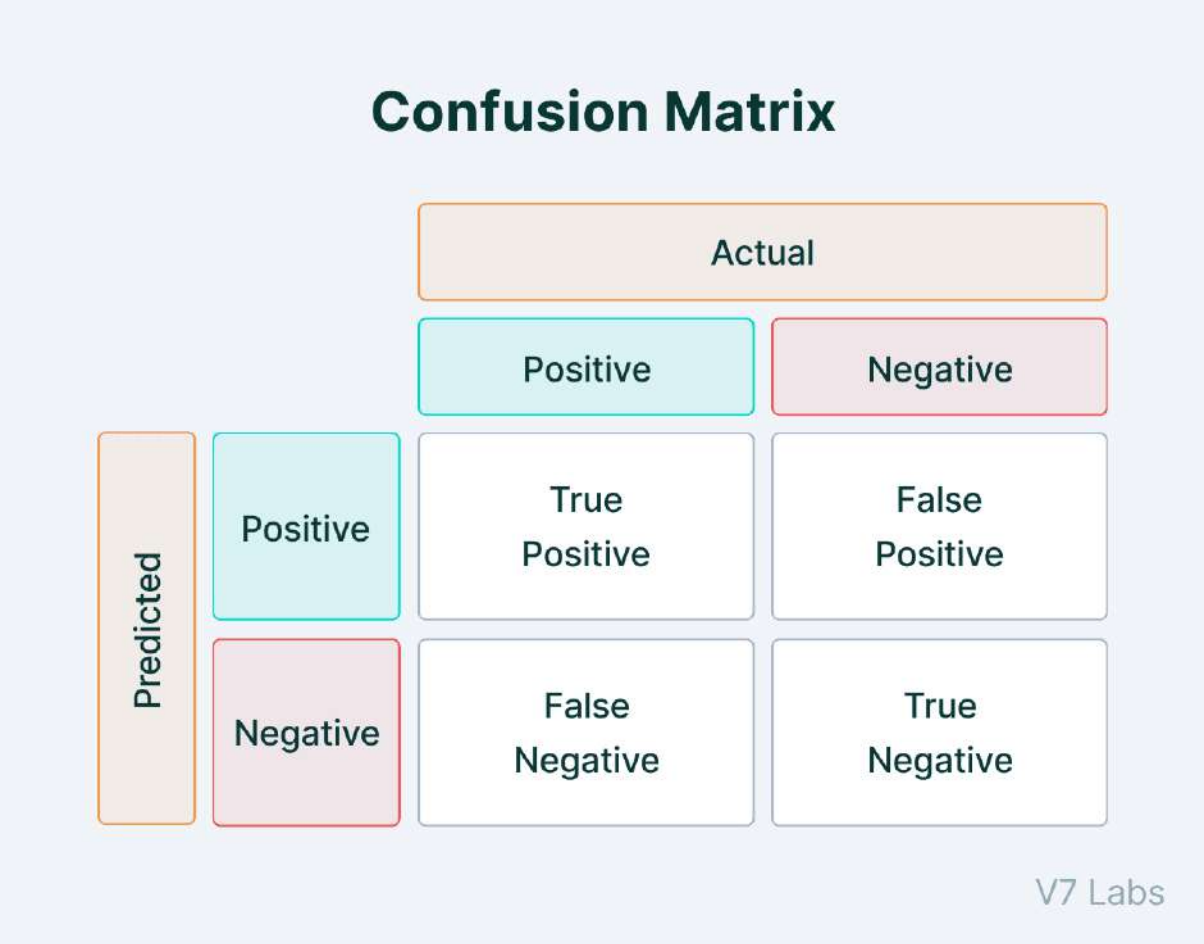


Figure 2.7 Illustration of a Confusion Matrix

Source: (Shah, 2022)

Where:

- True Positive (TP): The model predicts a label and matches correctly as ground truth
- True Negative (TN): The model correctly identifies that no object of that class is present.
- False Positive (FP): The model predicts a class where none exists in the ground truth.

- False Negative (FN): The model fails to predict a class that is actually in the ground truth.

In object detection, TN is generally omitted because background regions are not explicitly enumerated. Since most of the image consists of background pixels, counting TN would produce extremely large and uninformative values. For this reason, modern object detection evaluation relies only on TP, FP, and FN.

B. False Negative Rate

To strictly quantify the detection failures identified by the Confusion Matrix, this study employs the False Negative Rate (FNR). Conceptually, the false negative rate, also commonly referred to as the Miss Rate (MR) is defined as the probability that positive instance will be miss detected (Harness Team, 2025; Jiang et al., 2025).

In the domain of computer vision, this metric is established as a critical standard for evaluating detector reliability. Dollar et al. (2012) utilized Miss Rate as the primary benchmark for pedestrian detection, highlighting its importance over precision-based metrics when safety and coverage are paramount. Furthermore, recent frameworks for analyzing object detection failures utilize this metric to identify mechanisms where detectors fail to recognize target objects (Miller et al., 2022).

Mathematically, the metric is calculated as the ratio of false negatives to the total number of actual positive instances. As explicitly formulated in recent studies on object detection optimization (Jiang et al., 2025), the formula is expressed as:

$$MR = \frac{FN}{TP + FN} \quad (2.3)$$

This formulation is mathematically equivalent to the complement of Recall (1 - Recall), as formally derived in the Localization-Recall-Precision (LRP) error framework by Oksuz et al. (2018). A lower FNR value indicates a model with high sensitivity, capable of recovering fine-grained product variants that would otherwise be overlooked.

2.5.2 Object Detection Core Accuracy Evaluation Metrics

This subsection explains the four primary metrics used to evaluate object detection performance in this research: IoU, Precision–Recall, Average Precision (AP), and mean Average Precision (mAP). Each captures different aspects of localization quality and prediction accuracy.

A. Intersection over Union (IoU)

IoU measures the spatial alignment / overlap between the predicted bounding box coordinates and the ground-truth box. It is defined as the ratio between the area of intersection and the area of union of the two boxes:

$$\text{IoU} = \frac{|\text{Prediction} \cap \text{Ground Truth}|}{|\text{Prediction} \cup \text{Ground Truth}|} \quad (2.4)$$

A prediction is considered correct if its IoU exceeds a predefined threshold (e.g., 0.50 or 0.75). IoU serves as the foundational criterion for determining true positives and directly influences precision, recall, and the AP metrics used in this study.

Figure 2.8 Visual representation of Intersection over Union (IoU)

Source: (Shah, 2022)

B. Precision and Recall

Precision reflects the accuracy of the predictions, while recall reflects their completeness. Both precision (P) and recall (R) are formulated as follows:

$$P = \frac{TP}{TP + FP} \quad (2.5)$$

$$R = \frac{TP}{TP + FN} \quad (2.6)$$

Precision–recall relationships capture how the detector behaves at different confidence thresholds. These curves form the basis for computing Average Precision.

C. Average Precision (AP)

Average Precision (AP) measures detection quality per class by computing the area under the precision–recall curve. AP evaluates the detector's performance across all confidence thresholds, capturing trade-offs between precision and recall. AP is defined as:

$$AP(t) = \int_0^1 P(R, t)dR \quad (2.7)$$

Where:

- t represents IoU threshold used to define true positives
- $P(R, t)$ represents precision as a function of recall under IoU threshold t
- dR represents infinitesimal change in recall (continuous integration over the PR curve)

D. Mean Average Precision (mAP)

While AP is defined per class, object detection benchmarks typically measure performance across all classes. The mean of all AP values is called mean Average Precision (mAP). If a dataset contains n classes, mAP is computed as:

$$mAP = \frac{1}{n} \sum_{k=1}^{n} AP_k \quad (2.8)$$

Where:

- n represents number of classes
- AP_k represents AP value for class k

Generally, mAP reflects the detector's overall performance across all categories by treating each class equally. In modern evaluation protocols, particularly the COCO standard, the primary benchmark utilized is mAP50:95. Unlike simpler metrics, mAP50:95 computes the Average Precision (AP) across a range of Intersection over Union (IoU) thresholds from 0.50 to 0.95 in increments of 0.05, before averaging the results.

CHAPTER III
RESEARCH METHODOLOGY

3.1 Methodological Overview

The research methodology adopted in this study follows a structured sequence of stages that guide the development of the retail product detection model from start to finish. The overview of the entire methodology is presented in Figure 3.1.

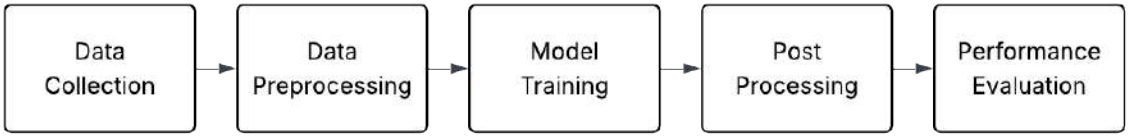


Figure 3.1 Methodology workflow

This study consists of five main stages. The first stage is data collection, in which retail product images are gathered from the confidential dataset and additional field observations. The second stage is data preprocessing, which includes data refinement, manual annotation, and dataset splitting. The third stage is model training, covering baseline model training, hyperparameter tuning, augmentation tuning, and dataset-level augmentation to obtain optimized detection models. The fourth stage is post-processing, where model inference outputs are adjusted and refined through threshold calibration and utilizing an ensemble technique, Weighted Boxes Fusion (WBF) to improve detection consistency. The fifth stage is performance evaluation, in which all model variants including baseline models, optimized models, and post-processed outputs are assessed using object detection metrics to enable fair and systematic comparison.

3.2 Data Collection

The dataset used in this research consists of retail shelf images focusing on product variants that share highly similar packaging designs but differ in grammage or size. The collected data are intended to represent real-world retail shelf conditions, where multiple visually similar product variants appear simultaneously within dense shelf layouts.

Figure 3.2 Example of an image collected for the data of the research

The dataset comprises 47 product classes grouped into 13 product families, where each class corresponds to a specific grammage or size variant within the same family. This class structure is designed to emphasize high intra-family visual similarity, which is the primary challenge addressed in this study.

The images capture common visual challenges encountered in retail environments, including variations in viewing angles, shelf arrangements, lighting conditions, and partial occlusions caused by neighboring products. These characteristics are preserved to ensure that the dataset reflects practical fine-grained detection scenarios.

3.3 Data Preprocessing

Data preprocessing is conducted to ensure that the dataset is consistent, standardized, and ready for the training phase. The preprocessing stage consists of three main steps, as shown in Figure 3.2.

Figure 3.3 Data preprocessing pipeline

3.3.1 Data Refinement

This stage focuses on refining the dataset by eliminating redundant images. Redundant and near-identical images introduce a critical bias in deep learning–based visual recognition by encouraging memorization rather than robust feature learning. Due to their high representational capacity, convolutional neural networks can easily exploit repeated or highly similar samples, resulting in overfitting and degraded generalization performance when confronted with unseen data (Aghabagherloo et al., 2025; Dipto et al., 2023). Empirical evidence shows that increasing image similarity within training data can significantly inflate classification accuracy, masking the true difficulty of the task and leading to unreliable model assessment (Dipto et al., 2023). This issue becomes more severe when near-identical samples appear across training and evaluation splits, constituting data leakage that artificially boosts reported performance metrics and undermines experimental validity (Barz & Denzler, 2020; Ramos et al., 2025). Beyond evaluation bias, redundant data also increases computational cost without contributing new discriminative information, as models repeatedly process visually indistinguishable patterns with minimal marginal learning benefit (CLRN team, 2025; John, 2025).

Figure 3.4 Image shows near-identical images present in the dataset.

Controlling redundancy through data refinement is a methodological necessity in deep learning workflows to ensure fair evaluation, stable training dynamics, and reliable conclusions. In this study, near-duplicate images typically captured from almost identical shelf arrangements or viewpoints were mostly removed to prevent leakage and over-representation of highly similar samples.

3.3.2 Data Labeling

The data labeling process was conducted manually by the researcher using the Roboflow annotation platform. Each image was annotated by drawing bounding boxes around visible product instances on retail shelves and assigning class labels corresponding to their respective product variants and grammage. This process ensures consistency with the predefined fine-grained class structure used throughout the study.

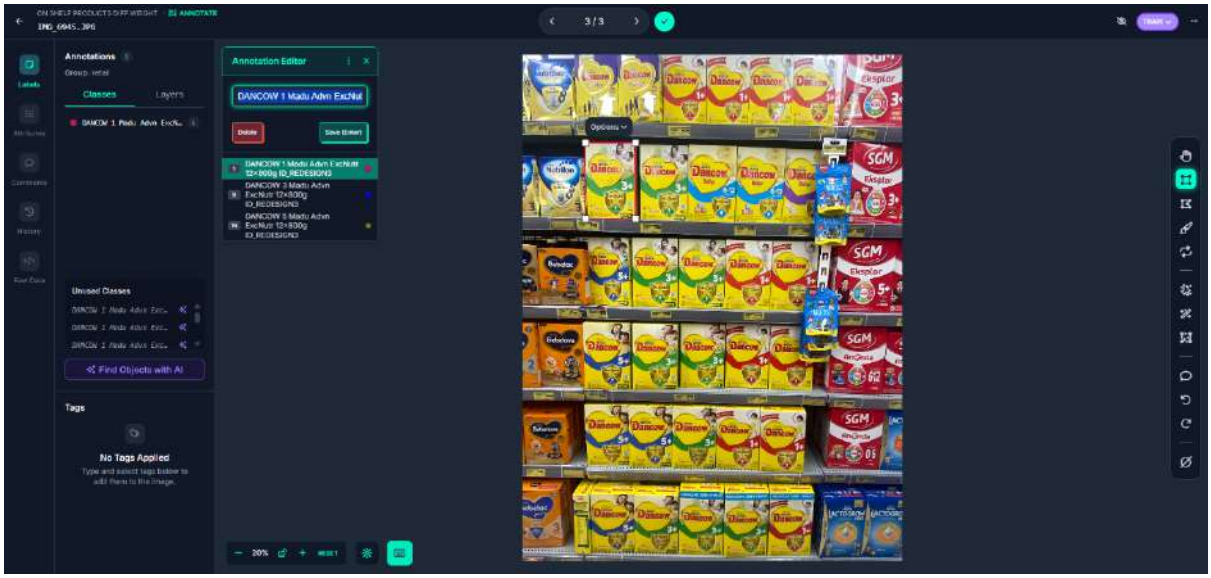


Figure 3.5 Image labeling process with Roboflow

After the labeling process was completed, all annotations were exported into two standardized formats to ensure compatibility with the detection frameworks used in this research. The first format follows the YOLO TXT structure, where each bounding box is represented using normalized coordinates (class, x_center, y_center, width, height). The second format follows the COCO JSON structure required for training the Faster R-CNN model.

3.3.3 Data Split

Before dividing the dataset into training and testing subsets, it is necessary to further address the issue of data leakage discussed in the previous subsection on data refinement. While refinement reduces leakage caused by redundant or near-duplicate images, leakage can still occur at the splitting stage if visually related images are distributed across subsets.

To reduce this risk, this research adopts a group-based splitting strategy. Images were first grouped according to their annotated class combinations, which implicitly organizes them by shelf scene and visual context. Each group was then assigned entirely to either the training or the testing subset, preventing the same shelf arrangement from appearing in both. This design conceptually aligns with the subject-wise split described in Rumala et al. (2023), where all samples originating from the same subject are kept within one fold to avoid identity-related leakage. In the context of retail shelf imagery, a shelf scene serves a similar role to a "subject," as images captured from the same arrangement share consistent spatial and visual patterns. This approach prevents visually coherent shelf scene contributes to both training and testing, thereby reducing the risk of soft leakage during evaluation.

In addition to the group-based split, this study also applied a difficulty-aware selection strategy. As the focus of this research is to distinguish products with nearly identical packaging but different weight variants, images containing two or more classes belonging to the same SKU family were considered as difficult samples. These difficult images typically feature multiple weight variants of the same product line (e.g., 1 kg, 750 g, 350 g, and 120 g variants appearing together), making them essential for evaluating fine-grained detection performance. To ensure a fair and representative evaluation, at least one difficult image for all classes was placed into the testing subset.

3.4 Model Training

The model training stage in this study consists of four main components. First, baseline training is conducted for both YOLOv12 and Faster R-CNN to establish reference performance under standard configurations. Second, a hyperparameter tuning process is performed to identify stable and high-performing learning-related parameters for each architecture. Third, an augmentation tuning stage is carried out to analyze the impact of on-the-fly data transformations on detection performance. Finally, a dataset-level optimization procedure is applied to enhance data diversity and fine-grained representativeness through synthetic sample generation.

To ensure a fair and consistent comparison across all optimization stages, model performance throughout the baseline, hyperparameter tuning, augmentation tuning, and dataset-level retraining phases is evaluated using the built-in validation pipeline provided by each respective training framework. In this stage, performance assessment is conducted directly on the validation set as part of the training process, rather than through separate inference-based evaluation. This unified validation-based protocol is adopted to avoid discrepancies arising from different post-processing pipelines and evaluation implementations, thereby ensuring that all reported improvements reflect genuine model and data optimization effects. All experiments were executed using NVIDIA Tesla V100 GPUs. The overall workflow of the model training stage is summarized in Figure 3.4.

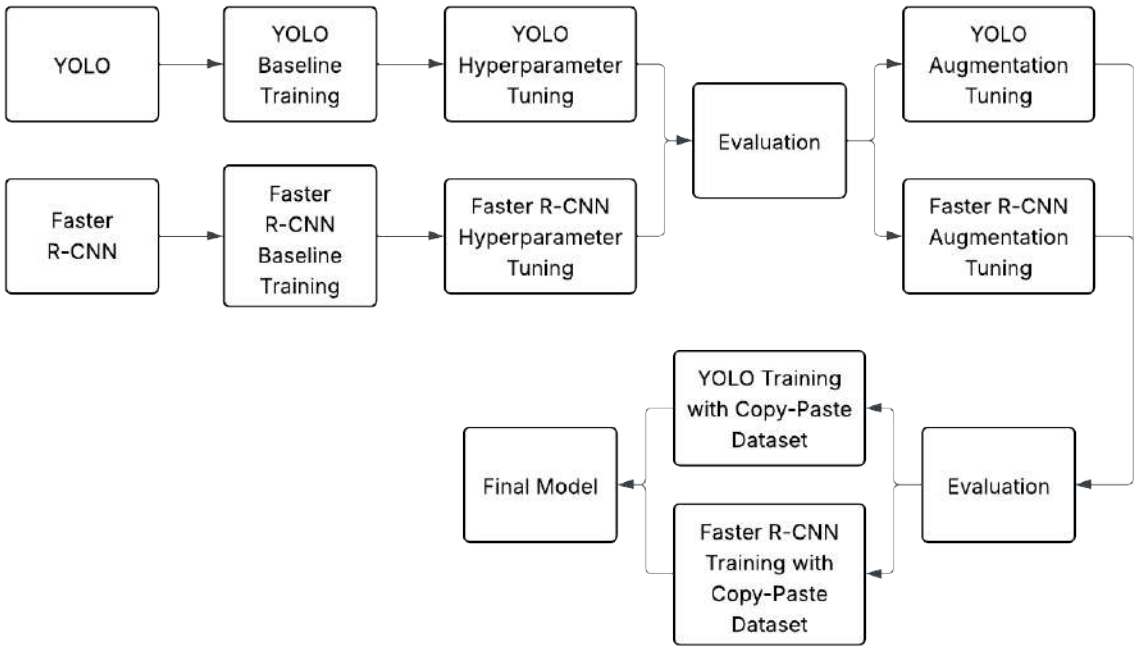


Figure 3.6 Model training stages

3.4.1 Baseline Model Training

During the baseline experiments, multiple variants of YOLOv12 (n, s, m, l, x) and Faster R-CNN (resnet50_fpn and resnet50_fpn_v2) were trained using identical configurations, and their performance was compared based on the resulting mAP scores. The highest-performing variant from each model family is selected as the backbone for subsequent optimization stages, while the remaining variants are excluded from further experimentation. Early stopping is also

applied to both experiments, as noted by Hussein and Shareef (2024), early stopping is an effective approach to reducing overfitting.

To establish the baseline, the models were initially configured using their automatic ('auto') or standard preset values to minimize manual intervention. This approach aligns with practices found in the documentation of each respective architecture (Jocher et al., 2023; Wu et al., 2019). Following preliminary exploratory trials to confirm training stability and convergence, the settings detailed in Table 3.1 and Table 3.2 were adopted as the baseline performance benchmark.

Table 3.1 BaselineYOLOv12 hyperparameter configuration

Hyperparameter	Value
Training epochs	100
Early stopping patience	20
Batch size	16
Image size	640
Optimizer	auto (default AdamW)
LR scheduler	auto (cosine default)
Warmup	auto
GPU count	2

Table 3.2 Baseline Faster R-CNN hyperparameter configuration

Hyperparameter	Value
Training epochs	150
Early stopping patience	25
Batch size	2
Image size	auto
Optimizer	SGD
LR scheduler	StepLR (step_size=20, gamma=0.1)
Learning rate	0.005
Momentum	0.9
Weight decay	0.0005
GPU count	1

During the baseline experiments, multiple variants of YOLOv12 (n, s, m, l, x) and Faster R-CNN (resnet50_fpn and resnet50_fpn_v2) were trained using identical configurations, and their performance was compared based on the resulting mAP scores. The highest-performing variant from each model family is then carried forward into the subsequent tuning stages, while the remaining variants are excluded from further optimization.

3.4.2 Hyperparameter Tuning

Hyperparameter tuning in this research is carried out using a hierarchical optimization strategy implemented with Optuna. The objective of this stage is to identify stable and high-performing configurations for both YOLOv12 and Faster R-CNN before proceeding to augmentation-level optimization. This hierarchical design ensures that learning-related parameters are optimized first, allowing subsequent augmentation tuning to be conducted on a strong and consistent foundation.

The first phase focuses exclusively on core hyperparameters that directly influence overall optimization behavior. Optuna is employed with the Tree-Structured Parzen Estimator (TPE) sampler, which adaptively explores promising regions of the search space while avoiding unnecessary evaluations. In this phase, the optimization search is executed over a predetermined number of trials: 50 trials for YOLOv12 and 25 trials for Faster R-CNN.

To accelerate the exploration process, core hyperparameter tuning is performed under a low-fidelity training budget. In this setup, models are trained for fewer epochs than their full training schedules: YOLOv12 is trained for 40 epochs (compared to its 100-epoch baseline), and Faster R-CNN is trained for 30 epochs (compared to its 150-epoch baseline). Using reduced-budget evaluations is a common and efficient approach in hyperparameter research, as early training behavior is often sufficient to differentiate promising configurations, even though full convergence is not yet achieved. Using a small number of training epochs is reasonable during early hyperparameter exploration, since even though early validation metrics are not fully reliable, they are still useful for identifying promising configurations (Franceschi et al., 2025).

Once all trials have been completed, the best-performing configuration for each model is selected and fixed. These optimized core hyperparameters then serve as the foundation for the augmentation-level tuning conducted in the next stage. The complete core hyperparameter search spaces for YOLOv12 and Faster R-CNN are summarized in Table 3.3 and Table 3.4.

Table 3.3 YOLOv12 hyperparameter search space

Hyperparameter	Value
Optimizer	{SGD, AdamW}
Initial learning rate (lr0)	5e-3 → 5e-2
Final learning rate factor (lrf)	0.005 → 0.05
Weight decay	1e-5 → 5e-4
Box loss gain	6.0 → 9.0
Class loss gain	0.2 → 0.8
Cosine learning rate (cos_lr)	{True, False}

Table 3.4 Faster R-CNN hyperparameter search space

Hyperparameter	Value
Learning rate	2e-3 → 1e-2
Momentum	0.85 → 0.95
Weight decay	1e-4 → 1e-3

3.4.3 Augmentation Tuning

Augmentation tuning is conducted after the core hyperparameters have been fixed, allowing this stage to focus solely on identifying the most effective data transformation strategies for each architecture. This phase aims to evaluate how different geometric, photometric, and compositional augmentations influence model performance, particularly in fine-grained retail product detection where subtle appearance variations must be learned robustly. This stage focuses only on on-the-fly augmentations applied during training.

To efficiently explore the augmentation space, Optuna is again employed as the optimization framework. In this stage, only augmentation-related parameters are varied, while all model-level hyperparameters remain locked to the best configurations obtained from the previous tuning phase. The augmentation optimization process is carried out over 25 trials for YOLOv12 and 20 trials for Faster R-CNN. Similar to core hyperparameter tuning, augmentation trials are executed under a reduced computational budget to accelerate experimentation while still revealing meaningful performance trends. For YOLOv12, each trial is trained for 50 epochs. For Faster R-CNN, each augmentation trial is trained for up to 8 epochs.

Augmentation tuning for Faster R-CNN is implemented using the Albumentations library, which provides a diverse set of transformations that preserve bounding-box geometry. The search space is restricted to augmentation types that are known to be safe for object detection tasks, ensuring that bounding boxes remain consistent after transformations. This includes

controlled variations in brightness, contrast, color, rotation angle, scaling, shifting, and noise probability.

Upon completion of all augmentation trials, augmentation configuration with the highest mAP for each model is selected. These optimized augmentation settings, together with the previously tuned core hyperparameters, form the complete configuration used for the final full-training stage. The augmentation search spaces for YOLOv12 and Faster R-CNN are summarized in Table 3.5 and Table 3.6.

Table 3.5 YOLOv12 augmentation search space

Augmentation Parameter	Value
HSV Hue (hsv_h)	0.00 → 0.03
HSV Saturation (hsv_s)	0.30 → 0.90
HSV Value (hsv_v)	0.20 → 0.90
Horizontal Flip (fliplr)	0.30 → 0.80
Vertical Flip (flipud)	0.00 → 0.20
Scale	0.40 → 0.90
Translation	0.00 → 0.15
Shear	0.00 → 0.15
Perspective	0.00 → 0.0008
Mosaic	0.50 → 1.00
MixUp	0.00 → 0.30

Table 3.6 Faster R-CNN augmentation search space

Augmentation Parameter	Value
Brightness	0.05 → 0.25
Contrast	0.05 → 0.25
Saturation	0.00 → 0.20
Hue	0.00 → 0.10
Rotation (degrees)	−8 → +8
Scale Limit	0.00 → 0.08
Shift Limit	0.00 → 0.08
Blur Probability	0.00 → 0.10
Noise Probability	0.00 → 0.10

3.4.4 Dataset-Level Augmentation and Model Retraining

Beyond on-the-fly augmentation applied during training, this study incorporates a dataset-level optimization strategy using Copy-Paste augmentation to further enhance the diversity and representativeness of the training data. Preliminary baseline experiments revealed that a significant portion of detection errors originated from visually similar product variants with limited training instances, particularly within the same product family and shelf tier. Based on this exploratory analysis, the Copy-Paste parameters were deliberately designed to be class-

aware and context-aware, prioritizing underrepresented and error-prone classes while preserving realistic spatial and semantic relationships. While online augmentation improves robustness at the pixel and geometric level, dataset-level augmentation enables explicit manipulation of object instances and backgrounds, thereby introducing new scene compositions that are not present in the original dataset.

```
for cls in TARGET_CLASSES:
    target_count = AUG_COUNTS[cls]
    crops = load_object_crops(cls)
    backgrounds = get_background_candidates(cls)

    generated = 0
    while generated < target_count:
        bg = random.choice(backgrounds)
        anchors = [o for o in bg["objects"] if o["group"] ==
CLASS_TO_GROUP[cls]]
        if not anchors:
            continue

        anchor = random.choice(anchors)
        crop = random.choice(crops)

        # scale crop relative to anchor width
        new_width = int((anchor["x2"] - anchor["x1"]) *
random.uniform(0.93, 1.07))
        scale = new_width / crop["width"]
        new_height = int(crop["height"] * scale)

        crop_resized = resize(crop["image"], new_width, new_height)

        # vertical alignment (same shelf tier)
        anchor_yc = (anchor["y1"] + anchor["y2"]) // 2
        yc = anchor_yc + random_jitter()

        # horizontal docking
        offset = random.randint(6, 12)
        if random_left():
            x2 = anchor["x1"] - offset
```

```
            x1 = x2 - new_width
        else:
            x1 = anchor["x2"] + offset
            x2 = x1 + new_width

        y1 = yc - new_height // 2
        y2 = y1 + new_height

        if not inside_image(bg["width"], bg["height"], x1, y1, x2, y2):
            continue

        new_box = (x1, y1, x2, y2)

        # overlap filtering
        reject = False
        for obj in bg["objects"]:
            _, ioa_old, _ = box_iou(obj["box"], new_box)
            if 0.1 < ioa_old < 0.7:
                reject = True
                break

        if reject:
            continue

        # paste and update labels
        paste_object(bg["image"], crop_resized, new_box)
        bg["labels"].append(build_yolo_label(cls,  new_box,  bg["width"],
bg["height"]))

        save_augmented_sample(bg)
        generated += 1
```

Figure 3.7 Code snippet of Copy-Paste data augmentation

Figure 3.7 presents a simplified algorithmic representation of the dataset-level Copy-Paste augmentation procedure. The algorithm iterates over a predefined set of target classes and generates a specified number of synthetic samples for each class. For every iteration, an object crop is selected and composited onto a compatible background image using an anchor-based placement strategy. Each accepted placement is then saved together with its automatically

generated annotation, and the process is repeated until the augmentation quota for the corresponding class is fulfilled

The implementation of this strategy was guided by a preliminary vulnerability analysis conducted during the exploratory phase, focusing on enriching scenarios identified as challenging in initial experiments. Crucially, the synthesis mechanism is designed to be additive: when a new object crop is pasted, the original object instances existing in the background image are preserved rather than discarded.

The resulting Copy-Paste augmented dataset is then used directly for full-fidelity model training using the previously optimized hyperparameter and augmentation configurations. The impact of dataset-level optimization is evaluated by comparing model performance trained on the original dataset and the Copy-Paste augmented dataset.

3.5 Post-Processing

The post-processing stage in this study is designed to refine raw model predictions prior to performance evaluation. Post-processing operates solely on inference outputs and aims to improve detection consistency across models and configurations. In this research, post-processing is primarily applied to support ensemble inference between YOLOv12 and Faster R-CNN.

3.5.1 Inference Output Preparation

For each experiment, YOLOv12 and Faster R-CNN are executed independently on the test dataset using their respective trained weights. Each model outputs a set of predicted bounding boxes, class labels, and confidence scores for every image.

Before further processing, inference-level parameters are adjusted to avoid premature filtering of detections. Confidence thresholds and IoU-related filtering are relaxed compared to default inference settings, allowing low to medium-confidence predictions to be retained. This adjustment is intended to preserve a richer set of bounding box candidates, which is typically beneficial for the subsequent Weighted Box Fusion stage, as WBF often achieves better fusion performance when provided with a broader range of detection hypotheses.

All predictions are exported in a unified format containing normalized bounding box coordinates, confidence scores, and class indices. This standardized output format ensures that predictions from YOLOv12 and Faster R-CNN can be processed consistently in the subsequent fusion stage.

3.5.2 WBF Ensemble Model

After inference outputs are prepared, predictions from YOLOv12 and Faster R-CNN are combined using a Weighted Boxes Fusion (WBF) procedure. The fusion process operates directly on the retained inference outputs and does not alter the internal behavior of either detector.

In this study, WBF is applied after exploring different inference-level configurations. Several WBF parameters, such as the IoU threshold for box grouping and confidence-related constraints are adjusted through controlled experiments to examine their impact on detection quality. This step is necessary because overly strict fusion settings may discard useful detections, while overly permissive settings may amplify false positives, particularly in dense shelf environments.

During fusion, bounding boxes predicted by YOLOv12 and Faster R-CNN that spatially overlap and share the same class label are grouped together. For each group, a single fused bounding box is generated by averaging the coordinates of contributing boxes, weighted by their confidence scores. The resulting fused prediction represents a consolidated detection that integrates spatial and confidence information from both models.

3.6 Performance Evaluation

The performance evaluation stage is designed to assess model behavior in a detailed and interpretable manner across all experimental configurations. Unlike the evaluation protocol applied during the model training stage which relies on validation-based assessment integrated into the training process, this stage evaluates model performance using inference-based predictions on the held-out test dataset. Evaluation is performed consistently for baseline models, optimized models, and ensemble of both models to ensure fair comparison under a unified inference pipeline.

3.6.1 Misclassification Analysis

Evaluation begins with an analysis of detection failures and misclassification patterns. For each model configuration, a confusion matrix is generated using predictions on the test dataset. This matrix aligns predicted labels with their corresponding ground truth, detailing the count of True Positives (TP), False Positives (FP), and False Negatives (FN) for each class.

To strictly quantify these tendencies, this study utilizes the False Negative Rate (FNR), or Miss Rate, as the primary diagnostic metric. FNR focuses on the completeness of the detection

relative to the ground truth, measuring the probability that a valid product instance is missed. This quantitative assessment allows this study to verify whether the optimization stages genuinely reduce the rate of missed detections among visually similar grammage variants.

3.6.2 Overall Accuracy Analysis

Following the misclassification analysis, performance is evaluated quantitatively using per-class Average Precision (AP) based on the COCO evaluation protocol. AP is computed across IoU thresholds ranging from 0.50 to 0.95 (AP50:95) to capture both localization accuracy and classification reliability. Per-class AP analysis provides a fine-grained view of model performance by revealing which product variants are consistently well detected and which remain challenging due to subtle visual differences or limited instance counts.

Finally, overall detection performance is summarized using mean Average Precision (mAP50:95) to enable global comparison across experimental stages. This evaluation strategy ensures that improvements reported are grounded in observable changes in model behavior, particularly in reducing grammage confusion among visually similar retail products.

CHAPTER IV
RESULTS AND DISCUSSIONS

4.1 Experimental Setup Overview

This section first describes the software environment and core libraries used throughout the experiments. All experiments were conducted using Python-based implementations with dedicated virtual environments for different detection architectures.

Table 4.1 Experimental software environment

Library / Framework	Version
python	3.10
ultralytics (YOLO)	8.3.225
torchvision (Faster R-CNN)	0.20.1+cu121
optuna	3.6.1
albumentations	2.0.8
ensemble-boxes	1.0.9

In addition, all experiments reported in this chapter are conducted using the final refined dataset obtained after the preprocessing stage. The finalized baseline dataset consists of 408 images containing 2,614 annotated object instances, which were manually labeled by the researcher as bounding boxes, across 47 fine-grained retail product classes, each corresponding to a specific product and grammage variant.

The dataset is divided into training and testing subsets using a group-based splitting strategy. The split contains 273 training images (66.91%) with 1,579 bounding boxes and 135 testing images (33.09%) with 1,035 bounding boxes. This configuration ensures that evaluation results reflect generalization to unseen shelf scenes rather than memorization of similar visual layouts.

Table 4.2 presents the distribution of annotated object instances across the 47 fine-grained product classes used in this study. Each class represents a distinct combination of product brand, variant, and packaging grammage. The distribution highlights the inherent imbalance across classes, which is common in fine-grained retail datasets where certain product variants appear more frequently on store shelves than others.

Table 4.2 List of classes

Product Family	ID	Class Name	Instances
DANCOW 1+ Madu	0	DANCOW 1+ Madu 1kg	52
	1	DANCOW 1+ Madu 750g	72
	2	DANCOW 1+ Madu 350g	99
	3	DANCOW 1+ Madu 120g	105
DANCOW 1+ Vanilla	4	DANCOW 1+ Van 1kg	49
	5	DANCOW 1+ Van 750g	105
	6	DANCOW 1+ Van 350g	107
	7	DANCOW 1+ Van 120g	42
DANCOW 3+ Madu	8	DANCOW 3+ Madu 1kg	85
	9	DANCOW 3+ Madu 750g	67
	10	DANCOW 3+ Madu 350g	85
DANCOW 3+ Vanilla	11	DANCOW 3+ Van 1kg	90
	12	DANCOW 3+ Van 750g	74
	13	DANCOW 3+ Van 350g	92
DANCOW 5+ Madu	14	DANCOW 5+ Madu 1kg	39
	15	DANCOW 5+ Madu 750g	94
	16	DANCOW 5+ Madu 350g	144
LACTOGEN 1 Happynutri	17	LACTOGEN 1 Happynutri 1kg	34
	18	LACTOGEN 1 Happynutri 735g	31
	19	LACTOGEN 1 Happynutri 350g	38
	20	LACTOGEN 1 Happynutri 180g	34
LACTOGEN 2 Happynutri	21	LACTOGEN 2 Happynutri 1kg	26
	22	LACTOGEN 2 Happynutri 735g	34
	23	LACTOGEN 2 Happynutri 350g	55
	24	LACTOGEN 2 Happynutri 180g	35
LACTOGEN PRO 0-6 mo	25	LACTOGEN PRO 0-6 mo 1kg	54
	26	LACTOGEN PRO 0-6 mo 735g	69
	27	LACTOGEN PRO 0-6 mo 350g	47
	28	LACTOGEN PRO 0-6 mo 180g	56
LACTOGEN PRO 6-12 mo	29	LACTOGEN PRO 6-12 mo 1kg	57
	30	LACTOGEN PRO 6-12 mo 735g	43
	31	LACTOGEN PRO 6-12 mo 350g	44
	32	LACTOGEN PRO 6-12 mo 180g	17
LACTOGROW 3 Happynutri Honey	33	LACTOGROW 3 Honey 1kg	29
	34	LACTOGROW 3 Honey 735g	36
	35	LACTOGROW 3 Honey 350g	36
	36	LACTOGROW 3 Honey 145g	26
LACTOGROW 3 Happynutri Vanilla	37	LACTOGROW 3 Van 1kg	14
	38	LACTOGROW 3 Van 735g	17
	39	LACTOGROW 3 Van 350g	29
LACTOGROW PRO 1+ Honey	40	LACTOGROW PRO 1+ Honey 1kg	14
	41	LACTOGROW PRO 1+ Honey 735g	25
	42	LACTOGROW PRO 1+ Honey 350g	84
	43	LACTOGROW PRO 1+ Honey 145g	36
LACTOGROW PRO 1+ Vanilla	44	LACTOGROW PRO 1+ Van 1kg	23
	45	LACTOGROW PRO 1+ Van 735g	25
	46	LACTOGROW PRO 1+ Van 350g	32

In addition to the prior refined base dataset, this study incorporates a Copy-Paste augmentation strategy to enhance data diversity. The augmented image captures a wider range of appearance variations than the original data alone. Figure 4.1 illustrates examples of instance crops as a part of the Copy-Paste process.

Figure 4.1 Copy-Paste instance crops

Figure 4.2 presents sample Copy-Paste results showing the insertion of additional an instance crop into shelf scenes. The augmented images are used only for the training subset.

(a) Original Image

(b) Copy-Paste Augmented Image

Figure 4.2 Comparison of original and augmented image

The augmentation process generated a total of 1,940 synthetic images. Crucially, since the augmentation mechanism preserves the original background objects while adding the targeted synthetic instances, this process resulted in a net addition of 16,128 object instances to the training set. This substantial expansion increased the total number of training annotations from the original 1,579 instances to a final count of 17,707 instances. The significant shift in class distribution and data volume specifically within the training subset resulting from this strategy is visualized in Figure 4.3.

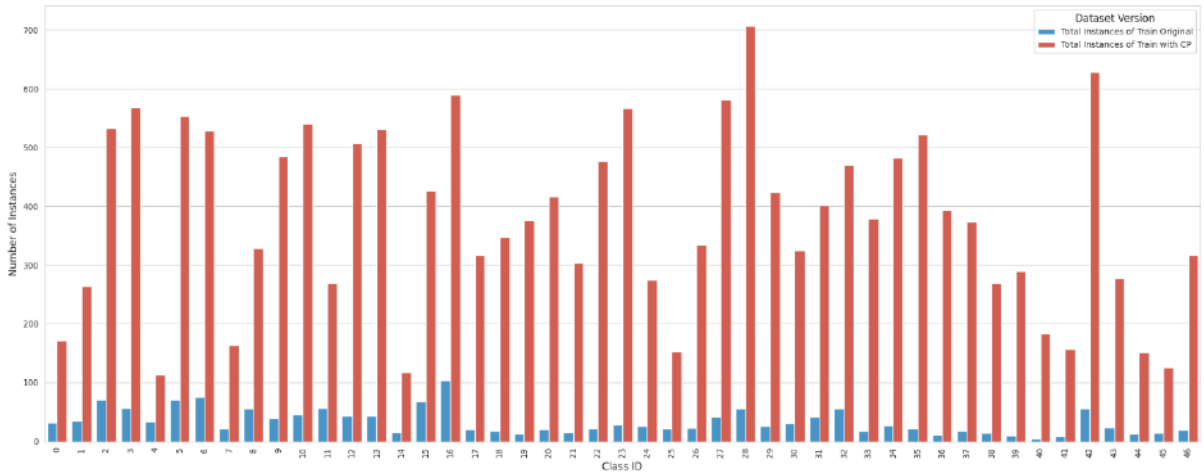


Figure 4.3 Total training data instances before and after augmentation

4.2 Progressive Optimization

The process begins with baseline model performance and proceeds through successive optimization stages, including core hyperparameter tuning, augmentation-level optimization, and dataset-level enhancement final model training. By reporting results at each stage, this section provides a transparent view of how individual optimization decisions contribute to improved fine-grained product discrimination and overall detection performance. This progressive evaluation ensures that observed performance gains are empirically grounded and directly attributable to specific optimization stages.

In all optimization stages reported in this section, model performance is evaluated using the default validation protocols provided by each detection framework during training. For YOLOv12, evaluation metrics are obtained from the built-in validation procedure executed at the end of each training epoch, while Faster R-CNN performance is measured using COCO-style evaluation on the validation split.

4.2.1 Baseline YOLOv12 Variants Comparison

To establish a reliable baseline for subsequent optimization stages, multiple YOLOv12 model variants with different architectural capacities are first evaluated. Specifically, the YOLOv12 *n*, *s*, *m*, *l*, and *x* variants are compared using identical training configurations. This comparison aims to analyze how model capacity influences detection performance in the context of fine-grained retail product recognition, where visual differences between classes are often subtle and highly localized.

In this baseline experiment, all YOLOv12 variants are trained using an identical configuration to ensure a fair comparison across different model capacities. The training is conducted for 100 epochs with early stopping patience of 20 epochs to prevent overfitting. A batch size of 16 and an input resolution of 640×640 are adopted as a balanced configuration between detection accuracy and computational efficiency. Pretrained weights are used to accelerate convergence and stabilize early training behavior. This configuration serves as the reference setting for evaluating the effect of architectural scaling. Figure 4.4 shows the code snippet of training command used for the YOLOv12x variant as an example.

```
yolo detect train \
    model=yolov12x.pt \
    data=data.yaml \
    epochs=100 \
    patience=20 \
    batch=16 \
    imgsz=640 \
    pretrained=true \
    seed=42 \
    deterministic=true
```

Figure 4.4 Code snippet of baseline training command for YOLOv12x

The same training configuration is applied to all YOLOv12 variants, with only the model backbone changed accordingly. This protocol ensures that observed performance differences originate solely from architectural capacity rather than training configuration discrepancies. Then, the evaluation focuses on the mean Average Precision at IoU thresholds from 0.50 to 0.95 (mAP50:95), which provides a comprehensive assessment of both localization accuracy and classification robustness across varying levels of strictness. The results of this baseline comparison are summarized in Table 4.2.

Table 4.3 Baseline YOLOv12 performance across model variants

YOLOv12 Model Variant	mAP50:95
YOLOv12n	0.624
YOLOv12s	0.793
YOLOv12m	0.849
YOLOv12l	0.838
YOLOv12x	**0.850**

As shown in Table 4.3, detection performance increases as model capacity scales from the n to m variants, as reflected by the corresponding mAP50:95 values. Among the evaluated configurations, YOLOv12x records the highest mAP50:95 on the test set. YOLOv12x is used as the baseline model configuration for all subsequent optimization stages in this study.

4.2.2 Baseline Faster R-CNN Variants Comparison

To establish a baseline configuration for Faster R-CNN, two backbone variants available in the TorchVision framework are evaluated, namely Faster R-CNN ResNet-50 FPN (R50) and Faster R-CNN ResNet-50 FPN v2 (R50-FPN v2). Unlike YOLOv12, which provides multiple scalable model sizes, TorchVision Faster R-CNN implementations are limited to a small number of standardized backbones.

The Faster R-CNN baseline models are trained using a standard configuration based on the TorchVision implementation. The SGD optimizer is employed with a learning rate of 0.005, momentum of 0.9, and weight decay of 0.0005, which are commonly adopted settings for object detection tasks. A StepLR scheduler with a step size of 20 epochs is used to gradually reduce the learning rate and improve convergence stability. This configuration establishes a consistent baseline for evaluating the effect of backbone selection. The code snippet of the baseline ResNet-50 v2 variant training command is shown in the following Figure 4.5.

```
# Load Faster R-CNN backbone with COCO pretraining
model = fasterrcnn_resnet50_fpn_v2(weights="COCO_V1")
model.roi_heads.box_predictor = FastRCNNPredictor(
    model.roi_heads.box_predictor.cls_score.in_features,
    num_classes
)

# Optimizer and learning rate scheduler
optimizer = torch.optim.SGD(
    model.parameters(),
```

```
    lr=0.005,
    momentum=0.9,
    weight_decay=0.0005
)

lr_scheduler = torch.optim.lr_scheduler.StepLR(
    optimizer,
    step_size=20,
    gamma=0.1
)

# Training loop
for epoch in range(num_epochs):
    model.train()
    for images, targets in train_loader:
        loss_dict = model(images, targets)
        loss = sum(loss_dict.values())

        optimizer.zero_grad()
        loss.backward()
        optimizer.step()

    lr_scheduler.step()

    # Validation using COCO protocol
    map_val = evaluate(model, val_dataset)
```

Figure 4.5 Code snippet of baseline training command for Faster R-CNN r50v2

Both backbone variants are trained using the same parameter configurations and evaluation protocol. Performance is measured using mean Average Precision at IoU thresholds from 0.50 to 0.95 (mAP50:95), which serves as the primary metric for baseline comparison. For each model, the best-performing epoch based on validation mAP50:95 is selected. The baseline comparison results are summarized in Table 4.4.

Table 4.4 Baseline Faster R-CNN performance across backbone variants

Faster R-CNN Backbone	**mAP50:95**
ResNet-50 FPN	0.700
ResNet-50 FPN v2	**0.709**

As shown in Table 4.4, Faster R-CNN ResNet-50 FPN v2 attains a slightly higher mAP50:95 compared to the original ResNet-50 FPN backbone. Although the absolute performance difference is relatively small, the v2 variant reaches its best validation performance in fewer training epochs, as indicated by the early stopping behavior. Based on these baseline results, Faster R-CNN ResNet-50 FPN v2 is used as the baseline backbone configuration for subsequent optimization stages in this study.

4.2.3 YOLOv12 Hyperparameter Tuning

Hyperparameter optimization was performed to improve the baseline YOLOv12 model's ability to distinguish fine-grained product variants. The optimization process aims to identify a configuration that enhances variant-level discrimination while maintaining stable localization performance under dense retail shelf conditions.

The optimization was conducted with a total of 50 trials, exploring the predefined hyperparameter search space described in Chapter III. The optimization employed a low-fidelity evaluation setting, which enables efficient comparison across configurations but does not represent final model performance. The optimization focuses on high-impact parameters that directly influence optimization dynamics, including optimizer type, learning rate, learning rate decay factor, weight decay, and loss function weights. A reduced training budget of 40 epochs is used to enable efficient comparison across trials, while validation mAP50:95 is used as the optimization objective. Figure 4.6 shows the code snippet of the optimization.

```
def objective(trial):

    # Hyperparameter search space
    optimizer = trial.suggest_categorical("optimizer", ["SGD", "AdamW"])
    lr0 = trial.suggest_float("lr0", 5e-3, 5e-2, log=True)
    lrf = trial.suggest_float("lrf", 0.005, 0.05)
    weight_decay = trial.suggest_float("weight_decay", 1e-5, 5e-4,
log=True)
    box_gain = trial.suggest_float("box", 6.0, 9.0)
    cls_gain = trial.suggest_float("cls", 0.2, 0.8)
    cos_lr = trial.suggest_categorical("cos_lr", [True, False])

    # Reduced-budget training
    epochs = 40
```

```
    # Training command
    cmd = [
        "yolo", "detect", "train",
        "model=yolov12x.pt",
        "data=data.yaml",
        f"epochs={epochs}",
        "batch=16",
        "imgsz=640",
        f"optimizer={optimizer}",
        f"lr0={lr0}",
        f"lrf={lrf}",
        f"weight_decay={weight_decay}",
        f"box={box_gain}",
        f"cls={cls_gain}",
        f"cos_lr={cos_lr}",
        "seed=42",
        "deterministic=true"
    ]

    run_training(cmd)

    # Read validation mAP50-95 from training logs
    score = read_best_map()
    return score


study = optuna.create_study(direction="maximize")
study.optimize(objective, n_trials=50)

best_params = study.best_params
```

Figure 4.6 Code snippet of hyperparameter optimization for YOLOv12 using Optuna

Out of the evaluated trials, one configuration consistently achieved the highest validation performance and was selected for subsequent full-fidelity training. Although the absolute validation mAP50:95 obtained at this stage is influenced by the low-fidelity setting, the selected configuration serves as a consistent reference point for further experiments.

Table 4.5 YOLOv12 selected hyperparameter configuration

Hyperparameter	Value
Optimizer	SGD
Initial learning rate (lr0)	0.005005
Final learning rate factor (lrf)	0.010289
Weight decay	0.00022999
Box loss weight	6.7359
Classification loss weight	0.6041
Cosine learning rate (cos_lr)	False

Compared with the baseline configuration trained under the same fidelity that achieved mAP50:95 of ≈0.833, the selected configuration obtained from this hyperparameter tuning stage achieved the highest validation mAP50:95 (≈0.85) among all evaluated trials and is used in all subsequent YOLOv12 experiments. Although this validation score does not represent the final evaluated performance, it provides a consistent basis for ranking candidate configurations under the same low-fidelity training setting. Further analysis of the impact of this configuration is presented in the following sections.

4.2.4 Faster R-CNN Hyperparameter Tuning

Hyperparameter optimization was applied to the Faster R-CNN baseline to improve its robustness in fine-grained product classification, particularly for closely related grammage variants. The optimization was carried out using Optuna with 25 trials, focusing on key training parameters including learning rate, momentum, and weight decay. Similar to YOLOv12, a low-fidelity evaluation setting was adopted to enable efficient exploration of the search space.

Hyperparameter tuning for Faster R-CNN is conducted by optimizing the core learning parameters that govern convergence behavior, namely learning rate, momentum, and weight decay. These parameters directly affect gradient updates and regularization strength. A reduced-budget training configuration is adopted to enable efficient exploration of the search space while maintaining consistent evaluation using the COCO mAP50:95 metric. Figure 4.7 shows the core hyperparameter optimization procedure used in this study.

```
def objective(trial):

    # Core hyperparameter search space
    learning_rate = trial.suggest_float("learning_rate", 2e-3, 1e-2,
log=True)
    momentum = trial.suggest_float("momentum", 0.85, 0.95)
```

```
    weight_decay = trial.suggest_float("weight_decay", 1e-4, 1e-3,
log=True)

    # Build Faster R-CNN model
    model = fasterrcnn_resnet50_fpn_v2(weights="COCO_V1")
    model.roi_heads.box_predictor = FastRCNNPredictor(
        model.roi_heads.box_predictor.cls_score.in_features,
        num_classes
    )

    optimizer = torch.optim.SGD(
        model.parameters(),
        lr=learning_rate,
        momentum=momentum,
        weight_decay=weight_decay
    )

    # Reduced-budget training
    for epoch in range(num_epochs):
        model.train()
        for images, targets in train_loader:
            loss_dict = model(images, targets)
            loss = sum(loss_dict.values())

            optimizer.zero_grad()
            loss.backward()
            optimizer.step()

        # Validation using COCO protocol
        map_val = evaluate(model, val_dataset)

    return map_val


study = optuna.create_study(direction="maximize")
study.optimize(objective, n_trials=25)

best_params = study.best_params
```

Figure 4.7 Code snippet of hyperparameter optimization for Faster R-CNN using Optuna

Among the evaluated trials, one configuration achieved the highest validation mAP50-95 and was selected for subsequent experiments. Despite the limited fidelity of this evaluation stage, this configuration shows higher validation performance than other evaluated parameter combinations. The selected hyperparameter values are summarized in Table 4.6.

Table 4.6 Faster R-CNN selected hyperparameter configuration

Hyperparameter	Value
Learning rate	0.00897
Momentum	0.9215
Weight decay	0.0002207

The selected hyperparameter configuration achieved the highest validation mAP50:95 (≈0.72) among all trials conducted under the low-fidelity evaluation setting and is used in all subsequent Faster R-CNN experiments, it performs better compared to the baseline configuration trained under the same fidelity in terms of the evaluated mAP (≈0.69). Similar to the YOLOv12 tuning stage, this validation score does not represent the final evaluated performance, but provides a consistent basis for comparing candidate configurations under identical conditions. Further analysis of its impact is presented in the following sections.

4.2.5 YOLOv12 Augmentation Tuning

Augmentation-level optimization was conducted to enhance the robustness of the YOLOv12 model under visual variations commonly encountered in retail shelf environments, including illumination changes, scale variation, spatial displacement, and partial occlusion. An Optuna-based search was employed to systematically explore combinations of color-space augmentation, geometric transformation, and composite augmentation strategies. A total of 25 augmentation trials were evaluated using a consistent low-fidelity training setup, with validation mAP50:95 serving as the optimization objective.

Augmentation tuning is performed to improve the robustness of YOLOv12 under visual variations commonly encountered in retail shelf environments. The optimization explores a combination of photometric augmentations in HSV color space, geometric transformations such as scaling, translation, and shear, and composite augmentations including Mosaic and MixUp. All core learning hyperparameters are fixed based on the previous tuning stage to ensure that the observed performance differences originate solely from the augmentation strategy. Figure 4.8 shows the augmentation optimization code snipppet.

```
def objective(trial):

    # Augmentation search space
    hsv_h = trial.suggest_float("hsv_h", 0.00, 0.03)
    hsv_s = trial.suggest_float("hsv_s", 0.30, 0.90)
    hsv_v = trial.suggest_float("hsv_v", 0.20, 0.90)

    fliplr = trial.suggest_float("fliplr", 0.30, 0.80)
    flipud = trial.suggest_float("flipud", 0.00, 0.20)

    scale       = trial.suggest_float("scale", 0.40, 0.90)
    translate   = trial.suggest_float("translate", 0.00, 0.15)
    shear       = trial.suggest_float("shear", 0.00, 0.15)
    perspective = trial.suggest_float("perspective", 0.00, 0.0008)

    mosaic = trial.suggest_float("mosaic", 0.50, 1.00)
    mixup  = trial.suggest_float("mixup", 0.00, 0.30)

    # Core hyperparameters are fixed (from previous tuning stage)
    epochs = 50

    cmd = [
        "yolo", "detect", "train",
        "model=yolov12x.pt",
        "data=data.yaml",
        f"epochs={epochs}",
        "batch=16",
        "imgsz=640",

        # Fixed core HP
        "optimizer=SGD",
        "lr0=0.005",
        "lrf=0.01",
        "momentum=0.937",
        "weight_decay=0.00023",

        # Tuned augmentation HP
        f"hsv_h={hsv_h}",
        f"hsv_s={hsv_s}",
        f"hsv_v={hsv_v}",
```

```
        f"fliplr={fliplr}",
        f"flipud={flipud}",
        f"scale={scale}",
        f"translate={translate}",
        f"shear={shear}",
        f"perspective={perspective}",
        f"mosaic={mosaic}",
        f"mixup={mixup}",

        "seed=42",
        "deterministic=true",
    ]

    run_training(cmd)

    # Read validation mAP50-95 from training logs
    score = read_best_map()
    return score


study = optuna.create_study(direction="maximize")
study.optimize(objective, n_trials=25)

best_aug_params = study.best_params
```

Figure 4.8 Code snippet of augmentation optimization for YOLOv12 using Optuna

After evaluating all augmentation trials under the same low-fidelity training budget, the configuration that achieves the highest validation mAP50:95 is selected as the optimized augmentation strategy for subsequent experiments. The selected augmentation parameters are summarized in Table 4.7.

Table 4.7 YOLOv12 selected augmentation configuration

Augmentation Parameter	Value
HSV Hue (hsv_h)	0.00025
HSV Saturation (hsv_s)	0.40626
HSV Value (hsv_v)	0.88491
Horizontal Flip (fliplr)	0.52999
Vertical Flip (flipud)	0.03354
Scale	0.58859
Translation	0.05084
Shear	0.10247
Perspective	0.00001
Mosaic	0.91505
MixUp	0.28814

Compared to the prior selected hyperparameter configuration (without augmentation tuning) trained under the same fidelity that achieved (≈0.666) mAP, this current stage's selected augmentation configuration achieved the highest validation mAP50:95 (≈0.856) among all evaluated trials and is used in subsequent YOLOv12 experiments.

4.2.6 Faster R-CNN Augmentation Tuning

Augmentation-level optimization was also applied to the Faster R-CNN baseline to improve its generalization capability under limited training data. In contrast to YOLOv12, augmentation for Faster R-CNN was implemented using Albumentations-based photometric and geometric transformations, focusing on brightness, contrast, saturation, spatial shifts, and mild image degradation. A search consisting of 20 trials was conducted, with validation mAP50:95 used as the optimization criterion.

For Faster R-CNN, augmentation tuning is implemented using Albumentations-based photometric and geometric transformations. The optimization focuses on brightness, contrast, saturation, hue, spatial shifts, scaling, rotation, and mild image degradation through blur and noise. These transformations are selected to simulate realistic variations in retail shelf imagery while preserving bounding-box consistency. Validation mAP50:95 is used as the optimization criterion. Figure 4.9 presents the augmentation optimization procedure used in this study.

```
def objective(trial):


    # Augmentation search space (Albumentations)
    brightness = trial.suggest_float("brightness", 0.05, 0.25)
    contrast   = trial.suggest_float("contrast",   0.05, 0.25)
```

```
    saturation = trial.suggest_float("saturation", 0.00, 0.20)
    hue        = trial.suggest_float("hue",        0.00, 0.10)

    rotate      = trial.suggest_int("rotate", -8, 8)
    scale_limit = trial.suggest_float("scale_limit", 0.00, 0.08)
    shift_limit = trial.suggest_float("shift_limit", 0.00, 0.08)

    blur_prob  = trial.suggest_float("blur_prob",  0.00, 0.10)
    noise_prob = trial.suggest_float("noise_prob", 0.00, 0.10)

    # Build augmentation pipeline
    aug = A.Compose([
        A.ColorJitter(brightness, contrast, saturation, hue, p=1.0),
        A.ShiftScaleRotate(shift_limit, scale_limit, rotate, p=1.0),
        A.GaussianBlur(blur_limit=(3,5), p=blur_prob),
        A.GaussNoise(var_limit=(5.0, 20.0), p=noise_prob),
    ], bbox_params=A.BboxParams(format="coco"))

    # Build Faster R-CNN model (core HP fixed)
    model = fasterrcnn_resnet50_fpn(weights="COCO_V1")
    model.roi_heads.box_predictor = FastRCNNPredictor(
        model.roi_heads.box_predictor.cls_score.in_features,
        num_classes
    )

    optimizer = torch.optim.AdamW(
        model.parameters(),
        lr=fixed_lr,
        weight_decay=fixed_weight_decay
    )

    # Reduced-budget training
    for epoch in range(num_epochs):
        train_one_epoch(model, train_loader, optimizer, aug)
        map_val = evaluate(model, val_dataset)

    return map_val


study = optuna.create_study(direction="maximize")
```

```
study.optimize(objective, n_trials=20)


best_aug_params = study.best_params
```

Figure 4.9 Code snippet of augmentation optimization for Faster R-CNN using Optuna

The augmentation configuration that achieves the highest validation mAP50:95 under the low-fidelity training setting is selected as the optimized augmentation strategy for Faster R-CNN. The selected augmentation parameters are summarized in Table 4.8.

Table 4.8 Faster R-CNN selected augmentation configuration

Augmentation Parameter	Value
Brightness	0.19625
Contrast	0.21095
Saturation	0.05447
Hue	0.00455
Rotation (degrees)	−3
Scale Limit	0.04886
Shift Limit	0.03784
Blur Probability	0.04004
Noise Probability	0.00166

The selected augmentation configuration achieved the highest validation mAP50:95 (≈0.415) among all evaluated trials and is used in subsequent Faster R-CNN experiments. However, as augmentation serves as a form of regularization that typically requires a longer convergence period to exhibit its full generalization benefits, a re-evaluation was conducted at 20 epochs to ensure a more representative comparison. Under this higher-fidelity setup, the prior Faster R-CNN tuned-hyperparameter configuration without augmentation achieved an mAP50:95 of 0.695. In contrast, the model integrated with the optimized augmentation configuration from this stage reached an mAP50:95 of 0.706.

4.2.7 YOLOv12 Dataset-Level Improvement: Copy-Paste Augmentation

This subsection evaluates the impact of dataset-level optimization on YOLOv12 performance through the application of Copy-Paste augmentation. Two training configurations are compared. The first model is trained using the original training dataset without additional dataset-level augmentation. The second model is trained using an expanded training dataset where Context-Aware Copy-Paste augmentation is applied to increase object instance diversity while preserving the same validation split. All other training settings, including model

architecture, optimized hyperparameters, and augmentation configurations obtained from the previous stages, are kept identical to ensure a controlled comparison.

In this experiment, dataset-level optimization is performed using Context-Aware Copy-Paste augmentation to increase object instance diversity and enrich underrepresented product variants. The model is retrained using the optimized hyperparameters and augmentation configuration obtained from the previous stages for 100 epochs. Figure 4.10 presents the final training configuration used for YOLOv12 with Copy-Paste–augmented data.

```
yolo detect train \
  model=yolov12x.pt \
  data=split_v2_cp/data.yaml \
  epochs=100 \
  batch=16 \
  imgsz=640 \
  optimizer=SGD \
  lr0=0.0050 \
  lrf=0.0103 \
  momentum=0.937 \
  weight_decay=0.00023 \
  box=6.74 \
  cls=0.60 \
  dfl=1.5 \
  warmup_epochs=3 \
  hsv_h=0.00025 \
  hsv_s=0.41 \
  hsv_v=0.88 \
  fliplr=0.53 \
  translate=0.05 \
  scale=0.59 \
  shear=0.10 \
  mosaic=0.92 \
  mixup=0.29 \
  auto_augment=randaugment \
  erasing=0.4 \
  seed=42 \
  deterministic=True
```

Figure 4.10 Code snippet of final retraining for YOLOv12 using augmented dataset

After retraining the model using both the original dataset and the Copy-Paste–augmented dataset under identical training conditions, performance is evaluated using the mAP50:95 metric. The comparison between the two dataset configurations is summarized in Table 4.9.

Table 4.9 Comparison of YOLOv12 with different training dataset

Training Configuration	mAP50:95
YOLOv12x (without Copy-Paste dataset augmentation)	0.867
YOLOv12x (with Copy-Paste dataset augmentation)	**0.889**

As shown in Table 4.9, the YOLOv12x model trained using the Copy-Paste–augmented dataset achieves a higher mAP50:95 than the model trained on the original dataset under identical training conditions. The reported results reflect the performance difference observed between the two training configurations when dataset-level Copy-Paste augmentation is applied.

4.2.8 Faster R-CNN Dataset-Level Improvement: Copy-Paste Augmentation

This subsection reports the impact of dataset-level optimization on Faster R-CNN performance through the application of Copy-Paste augmentation. Two training setups are compared. The first model is trained using the original training dataset without dataset-level Copy-Paste augmentation. The second model is trained using an expanded training dataset generated with Copy-Paste augmentation applied at the dataset level. Apart from the dataset composition, all training configurations, model architecture, and evaluation procedures remain unchanged to ensure a fair comparison.

The final Faster R-CNN retraining stage is conducted using the Copy-Paste–augmented dataset and the fully optimized configuration obtained from previous tuning stages. The model is trained for 150 epochs using the SGD optimizer with tuned learning rate, momentum, and weight decay, together with the selected Albumentations augmentation pipeline. This configuration is designed to evaluate the cumulative impact of dataset-level optimization under a full training budget. Figure 4.11 presents the final training configuration used for Faster R-CNN with the Copy-Paste–augmented dataset.

```
# Build Faster R-CNN model (optimized backbone)
model = fasterrcnn_resnet50_fpn_v2(weights="COCO_V1")
```

```
model.roi_heads.box_predictor = FastRCNNPredictor(
    model.roi_heads.box_predictor.cls_score.in_features,
    num_classes
)

# Optimized optimizer (from core HP tuning)
optimizer = torch.optim.SGD(
    model.parameters(),
    lr=0.00897,
    momentum=0.9215,
    weight_decay=0.00022
)

# Optimized augmentation (from augmentation tuning)
aug = A.Compose([
    A.ColorJitter(brightness=0.196,   contrast=0.211,   saturation=0.054,
hue=0.0045, p=1.0),
    A.ShiftScaleRotate(shift_limit=0.038,              scale_limit=0.049,
rotate_limit=-3, p=1.0),
    A.GaussianBlur(blur_limit=(3,5), p=0.04),
    A.GaussNoise(p=0.0016),
], bbox_params=A.BboxParams(format="coco"))

# Full-budget training
for epoch in range(150):
    train_one_epoch(model, train_loader, optimizer, aug)
    map_val = evaluate(model, val_dataset)

    if map_val > best_map:
        save_checkpoint(model)
```

Figure 4.11 Code snippet of final retraining for Faster R-CNN using augmented dataset

Following full-budget training on both the original and Copy-Paste–augmented datasets, model performance is evaluated using the mAP50:95 metric under identical evaluation conditions. The resulting performance comparison is summarized in Table 4.10.

Table 4.10 Comparison of Faster R-CNN with different training dataset

Training Configuration	mAP50:95
Faster R-CNN r50v2 (without Copy-Paste dataset augmentation)	0.753
Faster R-CNN r50v2 (with Copy-Paste dataset augmentation)	**0.849**

As shown in Table 4.10, the Faster R-CNN model trained using the Copy-Paste–augmented dataset records a higher mAP50:95 than the model trained on the original dataset under identical training conditions. The reported results summarize the performance differences observed between the two training setups when dataset-level Copy-Paste augmentation is applied. Based on this comparison, the Faster R-CNN model trained with the Copy-Paste–augmented dataset is used in subsequent experiments.

4.3 Performance Evaluation

This subsection presents the performance evaluation of the proposed detection framework with a primary emphasis on classification and its error analysis. The confusion matrix is used to examine false negative occurrences and misclassification patterns among fine-grained product classes, particularly between visually similar grammage variants. This analysis provides insight into error behavior that is not fully reflected by aggregate performance metrics.

Following the confusion matrix analysis, Average Precision (AP) is reported both at the per-class level and in aggregated form as mAP50:95 based on COCO evaluation, to quantitatively summarize detection performance across all classes.

4.3.1 Baseline YOLOv12x

The baseline YOLOv12 model is evaluated to establish its initial performance in detecting fine-grained product variants. This evaluation serves as a reference point for analyzing error patterns and performance limitations prior to the application of optimization strategies.

To analyze the distribution of detection errors across all product variants, a normalized confusion matrix is constructed based on the baseline YOLOv12x predictions on the test set. The confusion matrix visualizes the relationship between ground-truth labels and predicted classes, providing an overview of dominant confusion patterns among visually similar product variants. The resulting confusion matrix is presented in Figure 4.12.

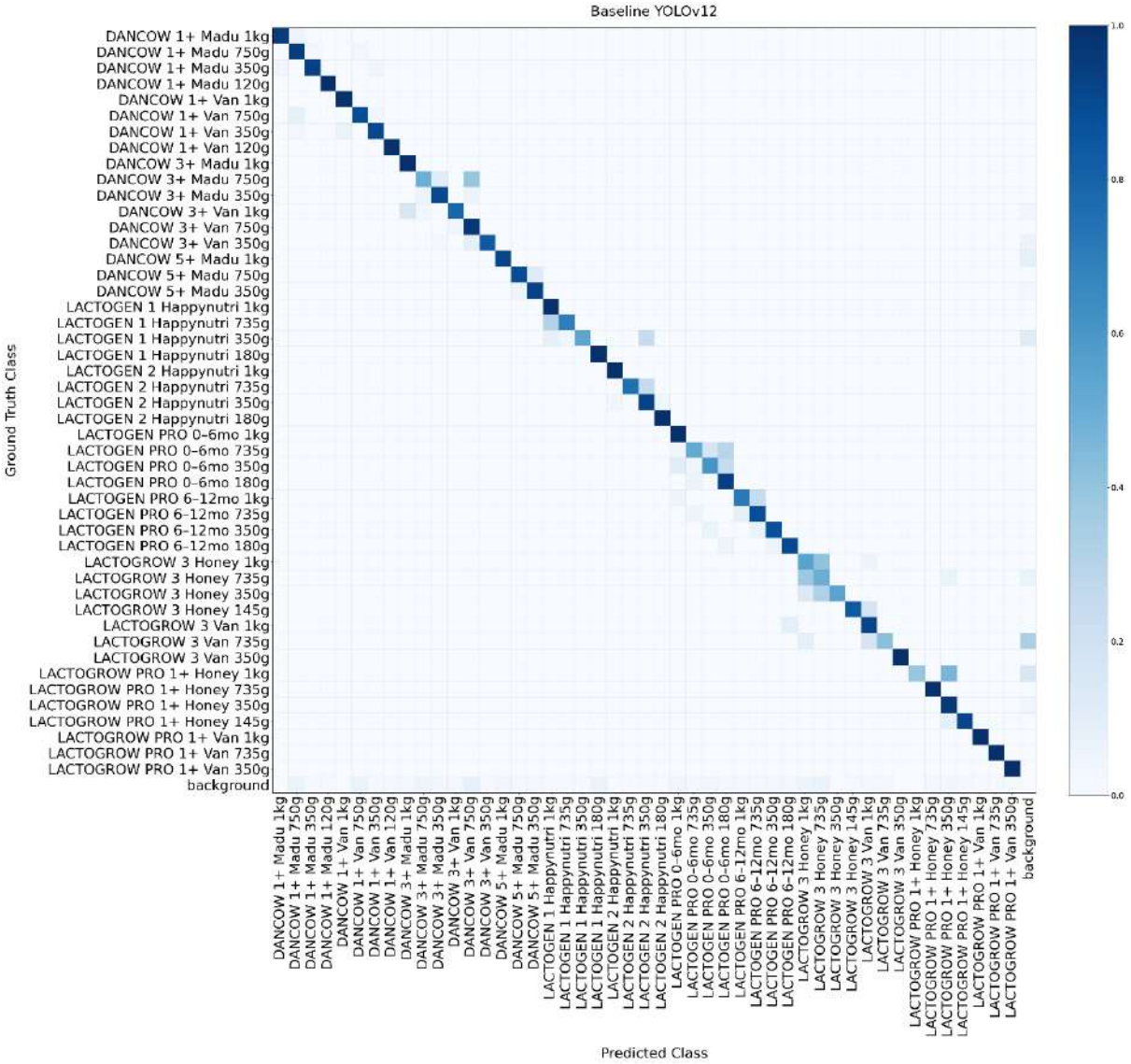


Figure 4.12 Baseline YOLOv12x confusion matrix

To complement the confusion matrix, a detailed misclassification breakdown is compiled for each product class. This breakdown reports the number of ground-truth instances, correctly detected instances, and the dominant misclassified classes along with their respective counts. The baseline misclassification statistics are summarized in Table 4.11.

Table 4.11 Baseline YOLOv12x misclassification breakdown

ID	Class / Product Name	GT	TP	FN / Misclassified by (Count)
0	DANCOW 1+ Madu 1kg	20	17	• DANCOW 1+ Madu 750g (1) • DANCOW 1+ Madu 120g (2)
1	DANCOW 1+ Madu 750g	37	33	• DANCOW 1+ Madu 350g (3) • DANCOW 1+ Van 750g (1)

2	DANCOW 1+ Madu 350g	29	26	• DANCOW 1+ Madu 750g (1) • DANCOW 1+ Van 350g (1) • DANCOW 3+ Madu 1kg (1)
3	DANCOW 1+ Madu 120g	48	45	• DANCOW 1+ Madu 350g (3)
4	DANCOW 1+ Van 1kg	15	14	• DANCOW 1+ Van 120g (1)
5	DANCOW 1+ Van 750g	35	28	• DANCOW 1+ Madu 750g (3) • DANCOW 1+ Madu 350g (1) • DANCOW 1+ Van 350g (1) • DANCOW 3+ Van 750g (2)
6	DANCOW 1+ Van 350g	32	26	• DANCOW 1+ Madu 750g (1) • DANCOW 1+ Madu 350g (1) • DANCOW 1+ Van 1kg (1) • DANCOW 1+ Van 120g (2) • DANCOW 3+ Madu 350g (1)
7	DANCOW 1+ Van 120g	20	15	• DANCOW 1+ Madu 350g (4) • DANCOW 1+ Van 1kg (1)
8	DANCOW 3+ Madu 1kg	29	18	• DANCOW 3+ Madu 750g (2) • DANCOW 3+ Van 1kg (8) • DANCOW 5+ Madu 1kg (1)
9	DANCOW 3+ Madu 750g	28	12	• DANCOW 3+ Madu 1kg (3) • DANCOW 3+ Madu 350g (1) • DANCOW 3+ Van 750g (12)
10	DANCOW 3+ Madu 350g	40	33	• DANCOW 3+ Madu 1kg (2) • DANCOW 3+ Madu 750g (2) • DANCOW 3+ Van 1kg (1) • DANCOW 3+ Van 750g (2)
11	DANCOW 3+ Van 1kg	33	27	• DANCOW 3+ Madu 1kg (4) • DANCOW 3+ Madu 750g (2)
12	DANCOW 3+ Van 750g	31	26	• DANCOW 3+ Madu 350g (1) • DANCOW 3+ Van 1kg (3 •)DANCOW 3+ Van 350g (1)
13	DANCOW 3+ Van 350g	49	45	• DANCOW 3+ Van 1kg (1) • DANCOW 3+ Van 750g (3)
14	DANCOW 5+ Madu 1kg	24	20	• DANCOW 3+ Madu 1kg (2) • DANCOW 5+ Madu 750g (2)
15	DANCOW 5+ Madu 750g	26	24	• DANCOW 3+ Madu 350g (1) • DANCOW 5+ Madu 350g (1)
16	DANCOW 5+ Madu 350g	41	37	• DANCOW 3+ Madu 350g (1) • DANCOW 5+ Madu 750g (2) • Background (1)
17	LACTOGEN 1 Happynutri 1kg	13	11	• LACTOGEN 1 Happynutri 735g (1) • LACTOGEN 2 Happynutri 1kg (1)
18	LACTOGEN 1 Happynutri 735g	13	10	• LACTOGEN 1 Happynutri 1kg (3)
19	LACTOGEN 1 Happynutri 350g	26	12	• LACTOGEN 1 Happynutri 1kg (4) • LACTOGEN 1 Happynutri 735g (4) • LACTOGEN 2 Happynutri 350g (6)
20	LACTOGEN 1 Happynutri 180g	13	11	• LACTOGEN 2 Happynutri 180g (2)

21	LACTOGEN 2 Happynutri 1kg	11	11	
22	LACTOGEN 2 Happynutri 735g	12	9	• DANCOW 3+ Madu 750g (1) • LACTOGEN 2 Happynutri 350g (1) • LACTOGEN 2 Happynutri 180g (1)
23	LACTOGEN 2 Happynutri 350g	26	21	• DANCOW 1+ Van 120g (1) • LACTOGEN 2 Happynutri 1kg (3) • LACTOGEN 2 Happynutri 180g (1)
24	LACTOGEN 2 Happynutri 180g	9	9	
25	LACTOGEN PRO 0-6 mo 1kg	12	9	• DANCOW 3+ Van 750g (1) • LACTOGEN PRO 0–6 mo 735g (2)
26	LACTOGEN PRO 0-6 mo 735g	31	20	• LACTOGEN PRO 0–6 mo 1kg (2) • LACTOGEN PRO 0–6 mo 350g (3) • LACTOGEN PRO 0–6 mo 180g (6)
27	LACTOGEN PRO 0-6 mo 350g	28	14	• LACTOGEN 2 Happynutri 735g (1) • LACTOGEN PRO 0–6 mo 1kg (2) • LACTOGEN PRO 0–6 mo 735g (9) • LACTOGEN PRO 0–6 mo 180g (2)
28	LACTOGEN PRO 0-6 mo 180g	33	28	• LACTOGEN PRO 0–6 mo 735g (1) • LACTOGEN PRO 0–6 mo 350g (4)
29	LACTOGEN PRO 6-12 mo 1kg	21	14	• LACTOGEN PRO 0–6 mo 735g (1) • LACTOGEN PRO 6–12 mo 735g (6)
30	LACTOGEN PRO 6-12 mo 735g	25	21	• LACTOGEN 1 Happynutri 1kg (1) • LACTOGEN PRO 6–12 mo 1kg (2) • LACTOGEN PRO 6–12 mo 180g (1)
31	LACTOGEN PRO 6-12 mo 350g	16	14	• LACTOGEN PRO 0–6 mo 350g (1) • LACTOGEN PRO 6–12 mo 735g (1)
32	LACTOGEN PRO 6-12 mo 180g	21	18	• LACTOGEN PRO 0–6 mo 180g (1) • LACTOGEN PRO 6–12 mo 350g (1) • LACTOGROW PRO 1+ Van 1kg (1)
33	LACTOGROW 3 Honey 1kg	20	9	• DANCOW 3+ Van 1kg (1) • DANCOW 3+ Van 350g (1) • LACTOGROW 3 Honey 735g (6) • LACTOGROW 3 Van 1kg (1) • LACTOGROW 3 Van 735g (2)
34	LACTOGROW 3 Honey 735g	16	10	• LACTOGEN PRO 0–6 mo 180g (1) • LACTOGROW 3 Honey 1kg (3) • LACTOGROW PRO 1+ Honey 350g (1) • Background (1)
35	LACTOGROW 3 Honey 350g	22	13	• DANCOW 1+ Madu 120g (1) • LACTOGEN PRO 0–6 mo 180g (1) • LACTOGROW 3 Honey 1kg (2) • LACTOGROW 3 Honey 735g (4) • LACTOGROW 3 Van 735g (1)
36	LACTOGROW 3 Honey 145g	6	5	• LACTOGROW 3 Van 1kg (1)
37	LACTOGROW 3 Van 1kg	12	9	• LACTOGEN PRO 6–12 mo 180g (1) • LACTOGROW 3 Honey 735g (1) • LACTOGROW 3 Van 735g (1)

38	LACTOGROW 3 Van 735g	12	7	• LACTOGROW 3 Honey 1kg (1) • LACTOGROW 3 Van 1kg (2) • LACTOGROW PRO 1+ Van 350g (2)
39	LACTOGROW 3 Van 350g	5	5	
40	LACTOGROW PRO 1+ Honey 1kg	13	5	• LACTOGROW PRO 1+ Honey 735g (2) • LACTOGROW PRO 1+ Honey 350g (6)
41	LACTOGROW PRO 1+ Honey 735g	8	5	• LACTOGROW PRO 1+ Honey 350g (3)
42	LACTOGROW PRO 1+ Honey 350g	28	27	• Background (1)
43	LACTOGROW PRO 1+ Honey 145g	12	10	• LACTOGROW PRO 1+ Honey 350g (2)
44	LACTOGROW PRO 1+ Van 1kg	11	11	
45	LACTOGROW PRO 1+ Van 735g	11	11	
46	LACTOGROW PRO 1+ Van 350g	12	12	

While the FP breakdown identifies the specific failure modes of the baseline model, the overall quantitative performance across all 47 classes is summarized using per-class Average Precision (AP50:95) values, presented in Table 4.12.

Table 4.12 YOLOv12x per-class average precision

ID	Class / Product Name	Average Precision (AP50:95)
0	DANCOW 1+ Madu 1kg	0.906692
1	DANCOW 1+ Madu 750g	0.913214
2	DANCOW 1+ Madu 350g	0.894216
3	DANCOW 1+ Madu 120g	0.924333
4	DANCOW 1+ Van 1kg	0.963051
5	DANCOW 1+ Van 750g	0.850063
6	DANCOW 1+ Van 350g	0.886939
7	DANCOW 1+ Van 120g	0.846656
8	DANCOW 3+ Madu 1kg	0.893866
9	DANCOW 3+ Madu 750g	0.382043
10	DANCOW 3+ Madu 350g	0.850283
11	DANCOW 3+ Van 1kg	0.743107
12	DANCOW 3+ Van 750g	0.831765
13	DANCOW 3+ Van 350g	0.793351
14	DANCOW 5+ Madu 1kg	0.849437
15	DANCOW 5+ Madu 750g	0.794135
16	DANCOW 5+ Madu 350g	0.905637
17	LACTOGEN 1 Happynutri 1kg	0.722853
18	LACTOGEN 1 Happynutri 735g	0.659255
19	LACTOGEN 1 Happynutri 350g	0.5496
20	LACTOGEN 1 Happynutri 180g	0.908003
21	LACTOGEN 2 Happynutri 1kg	0.962046
22	LACTOGEN 2 Happynutri 735g	0.688812

23	LACTOGEN 2 Happynutri 350g	0.829868
24	LACTOGEN 2 Happynutri 180g	0.949222
25	LACTOGEN PRO 0-6 mo 1kg	0.864822
26	LACTOGEN PRO 0-6 mo 735g	0.503280
27	LACTOGEN PRO 0-6 mo 350g	0.519592
28	LACTOGEN PRO 0-6 mo 180g	0.646778
29	LACTOGEN PRO 6-12 mo 1kg	0.640267
30	LACTOGEN PRO 6-12 mo 735g	0.750924
31	LACTOGEN PRO 6-12 mo 350g	0.799769
32	LACTOGEN PRO 6-12 mo 180g	0.7775
33	LACTOGROW 3 Honey 1kg	0.533219
34	LACTOGROW 3 Honey 735g	0.358615
35	LACTOGROW 3 Honey 350g	0.514649
36	LACTOGROW 3 Honey 145g	0.814851
37	LACTOGROW 3 Van 1kg	0.813402
38	LACTOGROW 3 Van 735g	0.296634
39	LACTOGROW 3 Van 350g	0.976238
40	LACTOGROW PRO 1+ Honey 1kg	0.333993
41	LACTOGROW PRO 1+ Honey 735g	0.897937
42	LACTOGROW PRO 1+ Honey 350g	0.876101
43	LACTOGROW PRO 1+ Honey 145g	0.785769
44	LACTOGROW PRO 1+ Van 1kg	0.9152
45	LACTOGROW PRO 1+ Van 735g	0.832384
46	LACTOGROW PRO 1+ Van 350g	0.864191

The baseline YOLOv12x evaluation establishes a reference detection profile for single-stage detectors on the fine-grained retail dataset. This baseline serves as a benchmark for assessing the effectiveness of subsequent optimization strategies and enables a direct comparison with two-stage detection architectures evaluated in the following subsection.

4.3.2 Baseline Faster R-CNN r50v2

The baseline Faster R-CNN r50v2 model is evaluated to assess its initial detection capability on the fine-grained retail product dataset. This evaluation provides a two-stage detection reference for comparison with the YOLOv12 baseline and establishes the initial error characteristics prior to optimization.

To analyze the distribution of detection errors produced by the two-stage detector, a normalized confusion matrix is constructed based on the Faster R-CNN r50v2 predictions on the test set. The confusion matrix summarizes the correspondence between ground-truth labels and predicted classes across all product variants. The resulting confusion matrix is shown in Figure 4.13.

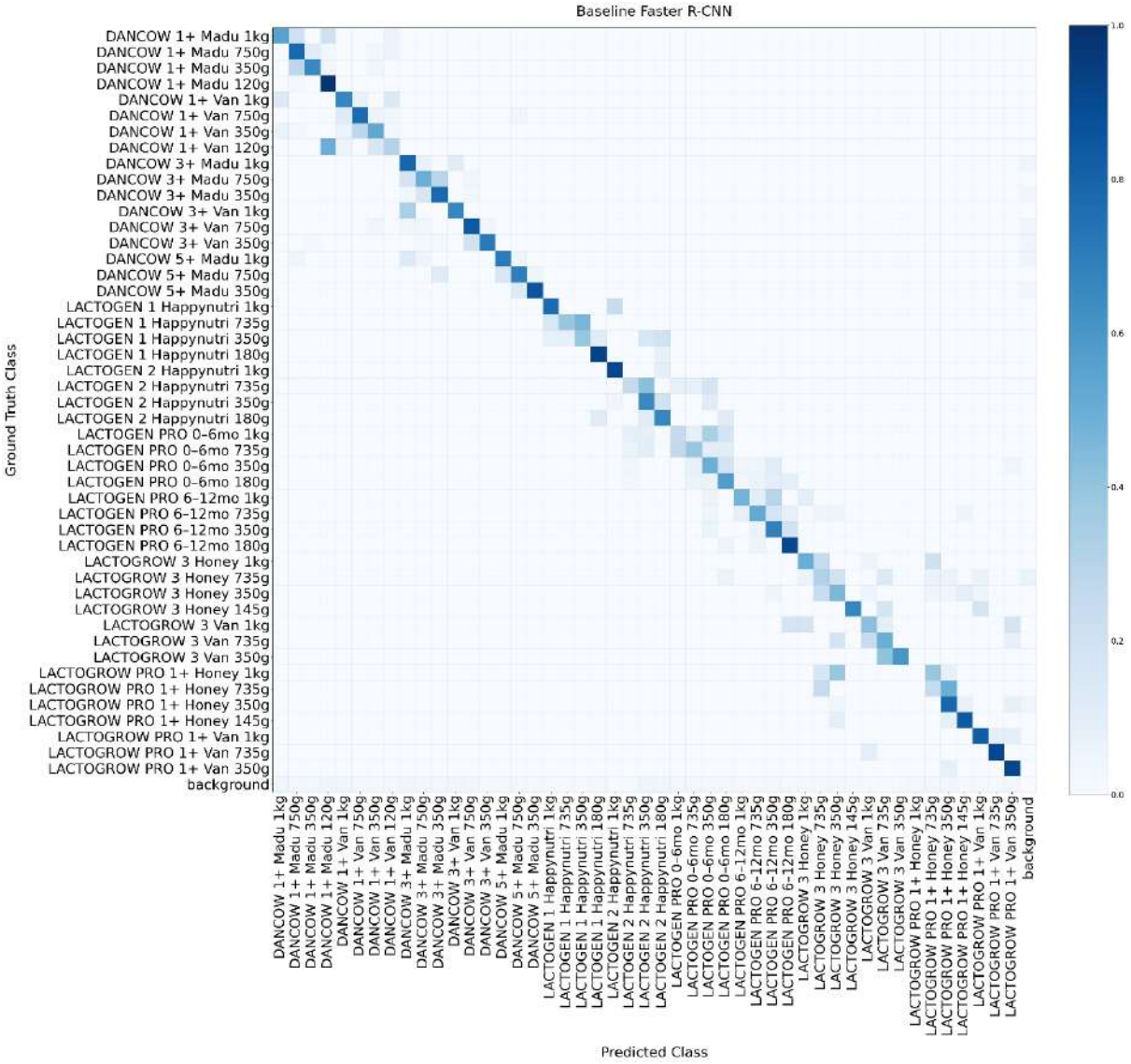


Figure 4.13 Baseline Faster R-CNN r50v2 confusion matrix

A detailed misclassification breakdown is further compiled to identify the dominant failure cases for each product variant. This table reports the number of ground-truth instances, true positives, and the most frequent misclassified classes for each SKU. The baseline Faster R-CNN misclassification statistics are summarized in Table 4.13.

Table 4.13 Baseline Faster R-CNN misclassification breakdown

ID	Class / Product Name	GT	TP	FN / Misclassified by (Count)
0	DANCOW 1+ Madu 1kg	20	11	• DANCOW 1+ Madu 750g (4) • DANCOW 1+ Madu 120g (4) • DANCOW 1+ Van 120g (1)

1	DANCOW 1+ Madu 750g	37	29	• DANCOW 1+ Madu 350g (4) • DANCOW 1+ Madu 120g (1) • DANCOW 1+ Van 350g (1) • DANCOW 1+ Van 120g (2)
2	DANCOW 1+ Madu 350g	29	19	• DANCOW 1+ Madu 750g (8) • DANCOW 1+ Madu 120g (1) • DANCOW 1+ Van 350g (1)
3	DANCOW 1+ Madu 120g	48	47	• DANCOW 1+ Madu 750g (1)
4	DANCOW 1+ Van 1kg	15	10	• DANCOW 1+ Madu 1kg (2) • DANCOW 1+ Van 750g (1) • DANCOW 1+ Van 120g (2)
5	DANCOW 1+ Van 750g	35	27	• DANCOW 1+ Van 1kg (4) • DANCOW 1+ Van 350g (2) • DANCOW 1+ Van 120g (1) • DANCOW 5+ Madu 750g (1)
6	DANCOW 1+ Van 350g	32	17	• DANCOW 1+ Madu 1kg (2) • DANCOW 1+ Madu 750g (1) • DANCOW 1+ Van 1kg (2) • DANCOW 1+ Van 750g (9) • DANCOW 1+ Van 120g (1)
7	DANCOW 1+ Van 120g	20	6	• DANCOW 1+ Madu 120g (10) • DANCOW 1+ Van 1kg (1) • DANCOW 1+ Van 350g (3)
8	DANCOW 3+ Madu 1kg	29	23	• DANCOW 3+ Madu 750g (2) • DANCOW 3+ Van 1kg (3) • Background (1)
9	DANCOW 3+ Madu 750g	28	14	• DANCOW 3+ Madu 1kg (5) • DANCOW 3+ Madu 350g (8) • DANCOW 3+ Van 750g (1)
10	DANCOW 3+ Madu 350g	40	31	• DANCOW 3+ Madu 1kg (1) • DANCOW 3+ Madu 750g (6) • DANCOW 3+ Van 750g (1) • Background (1)
11	DANCOW 3+ Van 1kg	33	22	• DANCOW 3+ Madu 1kg (11)
12	DANCOW 3+ Van 750g	31	26	• DANCOW 1+ Van 350g (1) • DANCOW 3+ Madu 1kg (1) • DANCOW 3+ Madu 750g (1) • DANCOW 3+ Van 350g (1) • Background (1)
13	DANCOW 3+ Van 350g	49	35	• DANCOW 1+ Madu 350g (1) • DANCOW 3+ Madu 750g (1) • DANCOW 3+ Madu 350g (1) • DANCOW 3+ Van 750g (9) • Background (2)
14	DANCOW 5+ Madu 1kg	24	17	• DANCOW 1+ Madu 750g (1) • DANCOW 3+ Madu 1kg (3) • DANCOW 3+ Madu 750g (1) • DANCOW 5+ Madu 750g (1) • Background (1)

15	DANCOW 5+ Madu 750g	26	18	• DANCOW 3+ Madu 350g (3) • DANCOW 5+ Madu 1kg (4) • DANCOW 5+ Madu 350g (1)
16	DANCOW 5+ Madu 350g	41	35	• DANCOW 5+ Madu 750g (5) • Background (1)
17	LACTOGEN 1 Happynutri 1kg	13	10	• LACTOGEN 2 Happynutri 1kg (3)
18	LACTOGEN 1 Happynutri 735g	13	5	• LACTOGEN 1 Happynutri 1kg (2) • LACTOGEN 1 Happynutri 350g (6)
19	LACTOGEN 1 Happynutri 350g	26	10	• LACTOGEN 1 Happynutri 1kg (2) • LACTOGEN 1 Happynutri 735g (2) • LACTOGEN 1 Happynutri 180g (3) • LACTOGEN 2 Happynutri 350g (4) • LACTOGEN 2 Happynutri 180g (5)
20	LACTOGEN 1 Happynutri 180g	13	12	• LACTOGEN 2 Happynutri 180g (1)
21	LACTOGEN 2 Happynutri 1kg	11	10	• LACTOGEN 2 Happynutri 180g (1)
22	LACTOGEN 2 Happynutri 735g	12	3	• LACTOGEN 2 Happynutri 350g (5) • LACTOGEN PRO 0-6 mo 1kg (1) • LACTOGEN PRO 0-6 mo 735g (1) • LACTOGEN PRO 0-6 mo 350g (2)
23	LACTOGEN 2 Happynutri 350g	26	17	• LACTOGEN 2 Happynutri 1kg (1) • LACTOGEN 2 Happynutri 180g (5) • LACTOGEN PRO 0-6 mo 350g (3)
24	LACTOGEN 2 Happynutri 180g	9	6	• LACTOGEN 1 Happynutri 180g (1) • LACTOGEN 2 Happynutri 350g (1) • LACTOGEN PRO 0-6 mo 180g (1)
25	LACTOGEN PRO 0-6 mo 1kg	12	3	• LACTOGEN 2 Happynutri 735g (1) • LACTOGEN 2 Happynutri 350g (1) • LACTOGEN PRO 0-6 mo 735g (1) • LACTOGEN PRO 0-6 mo 350g (4) • LACTOGEN PRO 0-6 mo 180g (2)
26	LACTOGEN PRO 0-6 mo 735g	31	12	• LACTOGEN 2 Happynutri 735g (2) • LACTOGEN 2 Happynutri 350g (3) • LACTOGEN PRO 0-6 mo 1kg (8) • LACTOGEN PRO 0-6 mo 350g (3) • LACTOGEN PRO 0-6 mo 180g (3)
27	LACTOGEN PRO 0-6 mo 350g	28	14	• LACTOGEN 2 Happynutri 735g (1) • LACTOGEN PRO 0-6 mo 735g (2) • LACTOGEN PRO 0-6 mo 180g (5) • LACTOGEN PRO 6-12 mo 1kg (1) • LACTOGEN PRO 6-12 mo 735g (1) • LACTOGEN PRO 6-12 mo 350g (3) • LACTOGROW PRO 1+ Van 350g (1)

28	LACTOGEN PRO 0-6 mo 180g	33	19	• LACTOGEN 2 Happynutri 735g (1) • LACTOGEN PRO 0-6 mo 735g (2) • LACTOGEN PRO 0-6 mo 350g (2) • LACTOGEN PRO 6-12 mo 1kg (1) • LACTOGEN PRO 6-12 mo 735g (2) • LACTOGEN PRO 6-12 mo 350g (3) • LACTOGEN PRO 6-12 mo 180g (3)
29	LACTOGEN PRO 6-12 mo 1kg	21	10	• LACTOGEN PRO 0-6 mo 350g (1) • LACTOGEN PRO 6-12 mo 735g (2) • LACTOGEN PRO 6-12 mo 350g (6) • LACTOGROW 3 Honey 1kg (2)
30	LACTOGEN PRO 6-12 mo 735g	25	13	• LACTOGEN PRO 0-6 mo 350g (1) • LACTOGEN PRO 6-12 mo 1kg (2) • LACTOGEN PRO 6-12 mo 350g (4) • LACTOGEN PRO 6-12 mo 180g (2) • LACTOGROW 3 Honey 735g (1) • LACTOGROW 3 Honey 350g (1) • LACTOGROW PRO 1+ Honey 145g (1)
31	LACTOGEN PRO 6-12 mo 350g	16	11	• LACTOGEN PRO 0-6 mo 350g (1) • LACTOGEN PRO 6-12 mo 735g (1) • LACTOGEN PRO 6-12 mo 180g (3)
32	LACTOGEN PRO 6-12 mo 180g	21	19	• LACTOGEN PRO 0-6 mo 180g (1) • LACTOGEN PRO 6-12 mo 735g (1)
33	LACTOGROW 3 Honey 1kg	20	10	• LACTOGROW 3 Honey 735g (5) • LACTOGROW 3 Van 1kg (1) • LACTOGROW PRO 1+ Honey 735g (4)
34	LACTOGROW 3 Honey 735g	16	5	• LACTOGEN PRO 0-6 mo 180g (1) • LACTOGROW 3 Honey 1kg (1) • LACTOGROW 3 Honey 350g (3) • LACTOGROW 3 Van 735g (2) • LACTOGROW PRO 1+ Honey 735g (1) • LACTOGROW PRO 1+ Honey 350g (1) • LACTOGROW PRO 1+ Van 1kg (1) • Background (1)
35	LACTOGROW 3 Honey 350g	22	10	• LACTOGEN PRO 6-12 mo 350g (1) • LACTOGROW 3 Honey 735g (5) • LACTOGROW 3 Van 1kg (1) • LACTOGROW PRO 1+ Honey 735g (1) • LACTOGROW PRO 1+ Honey 350g (1) • LACTOGROW PRO 1+ Honey 145g (2) • LACTOGROW PRO 1+ Van 1kg (1)
36	LACTOGROW 3 Honey 145g	6	4	• LACTOGROW 3 Van 735g (1) • LACTOGROW PRO 1+ Van 1kg (1)
37	LACTOGROW 3 Van 1kg	12	5	• LACTOGEN PRO 6-12 mo 180g (2) • LACTOGROW 3 Honey 1kg (2) • LACTOGROW 3 Van 735g (1) • LACTOGROW PRO 1+ Van 350g (2)
38	LACTOGROW 3 Van 735g	12	6	• LACTOGROW 3 Honey 350g (2) • LACTOGROW 3 Van 1kg (3) • LACTOGROW PRO 1+ Van 350g (1)

39	LACTOGROW 3 Van 350g	5	3	• LACTOGROW 3 Van 735g (2)
40	LACTOGROW PRO 1+ Honey 1kg	13	0	• LACTOGROW 3 Honey 735g (2) • LACTOGROW 3 Honey 350g (5) • LACTOGROW PRO 1+ Honey 735g (5) • LACTOGROW PRO 1+ Honey 350g (1)
41	LACTOGROW PRO 1+ Honey 735g	8	2	• LACTOGROW 3 Honey 735g (2) • LACTOGROW PRO 1+ Honey 350g (4)
42	LACTOGROW PRO 1+ Honey 350g	28	22	• LACTOGROW 3 Honey 350g (1) • LACTOGROW PRO 1+ Honey 145g (2) • LACTOGROW PRO 1+ Van 350g (2) • Background (1)
43	LACTOGROW PRO 1+ Honey 145g	12	10	• LACTOGROW 3 Honey 350g (1) • LACTOGROW PRO 1+ Honey 350g (1)
44	LACTOGROW PRO 1+ Van 1kg	11	9	• LACTOGROW PRO 1+ Van 735g (1) • LACTOGROW PRO 1+ Van 350g (1)
45	LACTOGROW PRO 1+ Van 735g	11	10	• LACTOGROW 3 Van 1kg (1)
46	LACTOGROW PRO 1+ Van 350g	12	11	• LACTOGROW PRO 1+ Honey 350g (1)

To provide a comprehensive performance reference, the resulting AP50:95 values, reflecting the baseline detection accuracy for each fine-grained variant under a two-stage architecture, are detailed in Table 4.14.

Table 4.14 Faster R-CNN r50v2 per-class average precision

ID	Class / Product Name	Average Precision (AP50:95)
0	DANCOW 1+ Madu 1kg	0.823379
1	DANCOW 1+ Madu 750g	0.875272
2	DANCOW 1+ Madu 350g	0.810955
3	DANCOW 1+ Madu 120g	0.892975
4	DANCOW 1+ Van 1kg	0.798911
5	DANCOW 1+ Van 750g	0.880383
6	DANCOW 1+ Van 350g	0.787915
7	DANCOW 1+ Van 120g	0.488125
8	DANCOW 3+ Madu 1kg	0.750326
9	DANCOW 3+ Madu 750g	0.787830
10	DANCOW 3+ Madu 350g	0.848673
11	DANCOW 3+ Van 1kg	0.695414
12	DANCOW 3+ Van 750g	0.642353
13	DANCOW 3+ Van 350g	0.786168
14	DANCOW 5+ Madu 1kg	0.811794
15	DANCOW 5+ Madu 750g	0.762226
16	DANCOW 5+ Madu 350g	0.894134
17	LACTOGEN 1 Happynutri 1kg	0.889024
18	LACTOGEN 1 Happynutri 735g	0.695231
19	LACTOGEN 1 Happynutri 350g	0.439517
20	LACTOGEN 1 Happynutri 180g	0.830439
21	LACTOGEN 2 Happynutri 1kg	0.769783

22	LACTOGEN 2 Happynutri 735g	0.587921
23	LACTOGEN 2 Happynutri 350g	0.728593
24	LACTOGEN 2 Happynutri 180g	0.896920
25	LACTOGEN PRO 0-6 mo 1kg	0.687399
26	LACTOGEN PRO 0-6 mo 735g	0.401033
27	LACTOGEN PRO 0-6 mo 350g	0.512844
28	LACTOGEN PRO 0-6 mo 180g	0.569806
29	LACTOGEN PRO 6-12 mo 1kg	0.5954
30	LACTOGEN PRO 6-12 mo 735g	0.562216
31	LACTOGEN PRO 6-12 mo 350g	0.769026
32	LACTOGEN PRO 6-12 mo 180g	0.822969
33	LACTOGROW 3 Honey 1kg	0.557636
34	LACTOGROW 3 Honey 735g	0.414224
35	LACTOGROW 3 Honey 350g	0.541613
36	LACTOGROW 3 Honey 145g	0.769554
37	LACTOGROW 3 Van 1kg	0.662036
38	LACTOGROW 3 Van 735g	0.700037
39	LACTOGROW 3 Van 350g	0.940594
40	LACTOGROW PRO 1+ Honey 1kg	0.281160
41	LACTOGROW PRO 1+ Honey 735g	0.213585
42	LACTOGROW PRO 1+ Honey 350g	0.814563
43	LACTOGROW PRO 1+ Honey 145g	0.790774
44	LACTOGROW PRO 1+ Van 1kg	0.867717
45	LACTOGROW PRO 1+ Van 735g	0.867133
46	LACTOGROW PRO 1+ Van 350g	0.789661

The baseline Faster R-CNN evaluation provides a two-stage detection reference point under identical dataset and evaluation conditions. In the next subsection, the impact of hyperparameter tuning and augmentation strategies on detection performance is examined through a comparative analysis against the baseline configuration.

4.3.3 Optimized YOLOv12x

The optimized YOLOv12 model is evaluated to examine changes in error distribution and class-level performance following the application of hyperparameter tuning and augmentation strategies. The confusion matrix is used to analyze shifts in false-positive patterns relative to the baseline model, while per-class AP50:95 values provide a quantitative summary of detection performance across SKU variants.

Following the application of hyperparameter tuning and augmentation optimization, the YOLOv12x model is re-evaluated on the test set. A normalized confusion matrix is constructed to examine changes in the distribution of detection errors relative to the baseline model. The resulting confusion matrix is shown in Figure 4.14.

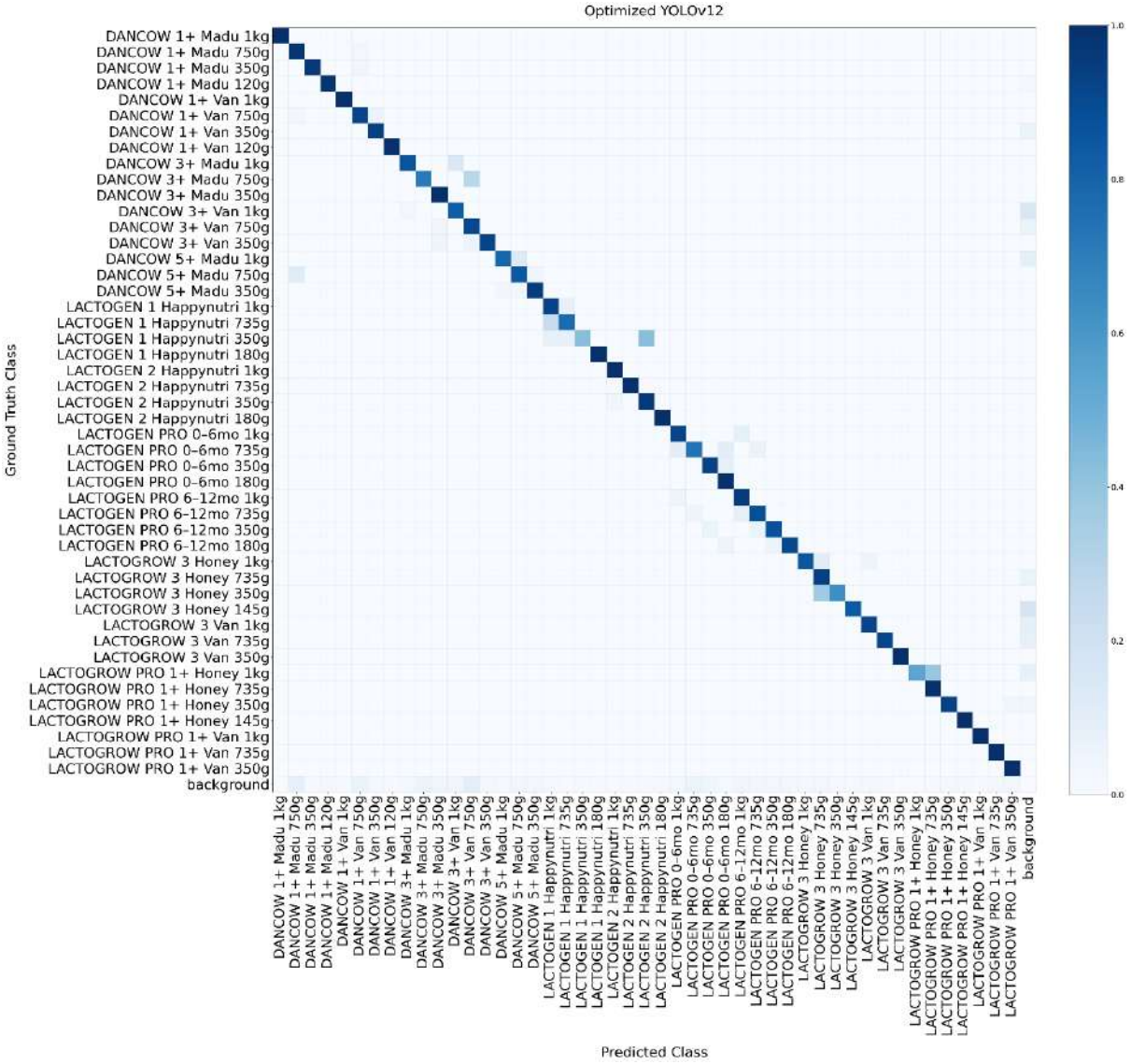


Figure 4.14 Optimized YOLOv12x confusion matrix

To provide a detailed view of class-level detection outcomes, a misclassification breakdown is compiled for the optimized YOLOv12x model. The table reports ground-truth counts, true positives, and the dominant misclassified classes for each product variant. The optimized misclassification statistics are summarized in Table 4.15.

Table 4.15 Optimized YOLOv12x misclassification breakdown

ID	Class / Product Name	GT	TP	FN / Misclassified by (Count)
0	DANCOW 1+ Madu 1kg	20	20	
1	DANCOW 1+ Madu 750g	37	36	• DANCOW 1+ Van 750g (1)
2	DANCOW 1+ Madu 350g	29	28	• DANCOW 1+ Van 750g (1)
3	DANCOW 1+ Madu 120g	48	47	• Background (1)
4	DANCOW 1+ Van 1kg	15	15	

5	DANCOW 1+ Van 750g	35	32	• DANCOW 1+ Madu 750g (1) • DANCOW 1+ Van 350g (2)
6	DANCOW 1+ Van 350g	32	30	• Background (2)
7	DANCOW 1+ Van 120g	20	20	
8	DANCOW 3+ Madu 1kg	29	25	• DANCOW 3+ Van 1kg (4)
9	DANCOW 3+ Madu 750g	28	20	• DANCOW 3+ Van 750g (8)
10	DANCOW 3+ Madu 350g	40	40	
11	DANCOW 3+ Van 1kg	33	27	• DANCOW 3+ Madu 1kg (1) • Background (5)
12	DANCOW 3+ Van 750g	31	28	• DANCOW 3+ Madu 350g (1) • Background (2)
13	DANCOW 3+ Van 350g	49	45	• DANCOW 3+ Madu 350g (2) • DANCOW 3+ Van 750g (2)
14	DANCOW 5+ Madu 1kg	24	19	• DANCOW 5+ Madu 750g (3) • Background (2)
15	DANCOW 5+ Madu 750g	26	22	• DANCOW 1+ Madu 750g (3) • DANCOW 5+ Madu 350g (1)
16	DANCOW 5+ Madu 350g	41	39	• DANCOW 5+ Madu 1kg (1) • DANCOW 5+ Madu 750g (1)
17	LACTOGEN 1 Happynutri 1kg	13	12	• LACTOGEN 1 Happynutri 735g (1)
18	LACTOGEN 1 Happynutri 735g	13	10	• LACTOGEN 1 Happynutri 1kg (3)
19	LACTOGEN 1 Happynutri 350g	26	11	• LACTOGEN 1 Happynutri 1kg (2) • LACTOGEN 1 Happynutri 735g (2) • LACTOGEN 2 Happynutri 350g (11)
20	LACTOGEN 1 Happynutri 180g	13	13	
21	LACTOGEN 2 Happynutri 1kg	11	11	
22	LACTOGEN 2 Happynutri 735g	12	12	
23	LACTOGEN 2 Happynutri 350g	26	25	• LACTOGEN 2 Happynutri 1kg (1)
24	LACTOGEN 2 Happynutri 180g	9	9	
25	LACTOGEN PRO 0-6 mo 1kg	12	11	• LACTOGEN PRO 6–12 mo 1kg (1)
26	LACTOGEN PRO 0-6 mo 735g	31	23	• LACTOGEN PRO 0–6 mo 1kg (3) • LACTOGEN PRO 0–6 mo 180g (3) • LACTOGEN PRO 6–12 mo 735g (2)
27	LACTOGEN PRO 0-6 mo 350g	28	26	• LACTOGEN PRO 0–6 mo 180g (2)
28	LACTOGEN PRO 0-6 mo 180g	33	33	
29	LACTOGEN PRO 6-12 mo 1kg	21	20	• LACTOGEN PRO 0–6 mo 1kg (1)
30	LACTOGEN PRO 6-12 mo 735g	25	22	• LACTOGEN PRO 0–6 mo 735g (1) • LACTOGEN PRO 6–12 mo 1kg (2)
31	LACTOGEN PRO 6-12 mo 350g	16	14	• LACTOGEN PRO 0–6 mo 350g (1) • LACTOGEN PRO 6–12 mo 735g (1)

32	LACTOGEN PRO 6-12 mo 180g	21	19	• LACTOGEN PRO 0–6 mo 180g (1) • LACTOGEN PRO 6–12 mo 350g (1)
33	LACTOGROW 3 Honey 1kg	20	17	• LACTOGROW 3 Honey 735g (2) • LACTOGROW 3 Van 1kg (1)
34	LACTOGROW 3 Honey 735g	16	15	• Background (1)
35	LACTOGROW 3 Honey 350g	22	14	• LACTOGROW 3 Honey 735g (8)
36	LACTOGROW 3 Honey 145g	6	5	• Background (1)
37	LACTOGROW 3 Van 1kg	12	11	• Background (1)
38	LACTOGROW 3 Van 735g	12	11	• Background (1)
39	LACTOGROW 3 Van 350g	5	5	
40	LACTOGROW PRO 1+ Honey 1kg	13	7	• LACTOGROW PRO 1+ Honey 735g (5) • Background (1)
41	LACTOGROW PRO 1+ Honey 735g	8	8	
42	LACTOGROW PRO 1+ Honey 350g	28	26	• LACTOGROW PRO 1+ Van 350g (1) • Background (1)
43	LACTOGROW PRO 1+ Honey 145g	12	12	
44	LACTOGROW PRO 1+ Van 1kg	11	11	
45	LACTOGROW PRO 1+ Van 735g	11	11	
46	LACTOGROW PRO 1+ Van 350g	12	12	

Following the observed shift in error distribution, the impact of hyperparameter and augmentation optimization on overall variant discrimination is quantitatively assessed using the per-class AP50:95 results, as summarized in Table 4.16.

Table 4.16 Optimized YOLOv12x per-class average precision

ID	Class / Product Name	Average Precision (AP50:95)
0	DANCOW 1+ Madu 1kg	0.966535
1	DANCOW 1+ Madu 750g	0.962045
2	DANCOW 1+ Madu 350g	0.926760
3	DANCOW 1+ Madu 120g	0.938022
4	DANCOW 1+ Van 1kg	0.951512
5	DANCOW 1+ Van 750g	0.927474
6	DANCOW 1+ Van 350g	0.957546
7	DANCOW 1+ Van 120g	0.900697
8	DANCOW 3+ Madu 1kg	0.831978
9	DANCOW 3+ Madu 750g	0.747254
10	DANCOW 3+ Madu 350g	0.938960
11	DANCOW 3+ Van 1kg	0.824741
12	DANCOW 3+ Van 750g	0.840179
13	DANCOW 3+ Van 350g	0.871333
14	DANCOW 5+ Madu 1kg	0.798336
15	DANCOW 5+ Madu 750g	0.955589
16	DANCOW 5+ Madu 350g	0.964414

17	LACTOGEN 1 Happynutri 1kg	0.812985
18	LACTOGEN 1 Happynutri 735g	0.774234
19	LACTOGEN 1 Happynutri 350g	0.514655
20	LACTOGEN 1 Happynutri 180g	0.896057
21	LACTOGEN 2 Happynutri 1kg	0.951591
22	LACTOGEN 2 Happynutri 735g	0.928789
23	LACTOGEN 2 Happynutri 350g	0.876642
24	LACTOGEN 2 Happynutri 180g	0.934653
25	LACTOGEN PRO 0-6 mo 1kg	0.799918
26	LACTOGEN PRO 0-6 mo 735g	0.756447
27	LACTOGEN PRO 0-6 mo 350g	0.973032
28	LACTOGEN PRO 0-6 mo 180g	0.928773
29	LACTOGEN PRO 6-12 mo 1kg	0.844773
30	LACTOGEN PRO 6-12 mo 735g	0.809469
31	LACTOGEN PRO 6-12 mo 350g	0.806365
32	LACTOGEN PRO 6-12 mo 180g	0.820059
33	LACTOGROW 3 Honey 1kg	0.790153
34	LACTOGROW 3 Honey 735g	0.884901
35	LACTOGROW 3 Honey 350g	0.691195
36	LACTOGROW 3 Honey 145g	0.805066
37	LACTOGROW 3 Van 1kg	0.963168
38	LACTOGROW 3 Van 735g	0.878811
39	LACTOGROW 3 Van 350g	0.972277
40	LACTOGROW PRO 1+ Honey 1kg	0.551565
41	LACTOGROW PRO 1+ Honey 735g	0.891667
42	LACTOGROW PRO 1+ Honey 350g	0.830268
43	LACTOGROW PRO 1+ Honey 145g	0.821178
44	LACTOGROW PRO 1+ Van 1kg	0.955671
45	LACTOGROW PRO 1+ Van 735g	0.892311
46	LACTOGROW PRO 1+ Van 350g	0.810816

The optimized YOLOv12x results demonstrate the effect of hyperparameter and augmentation tuning on single-stage detector performance. To provide a balanced comparison, the following subsection evaluates the optimized Faster R-CNN r50v2 model under the same dataset and evaluation protocol.

4.3.4 Optimized Faster R-CNN r50v2

The optimized Faster R-CNN model is evaluated to examine changes in error distribution and class-level performance following the application of hyperparameter tuning and proposal configuration adjustments. The confusion matrix is used to analyze shifts in false-positive patterns relative to the baseline model, while per-class AP50:95 values summarize detection performance across SKU variants.

After applying hyperparameter tuning and proposal configuration adjustments, the Faster R-CNN r50v2 model is re-evaluated on the test set. A normalized confusion matrix is

constructed to analyze the updated distribution of detection errors relative to the baseline configuration. The resulting confusion matrix is shown in Figure 4.15.

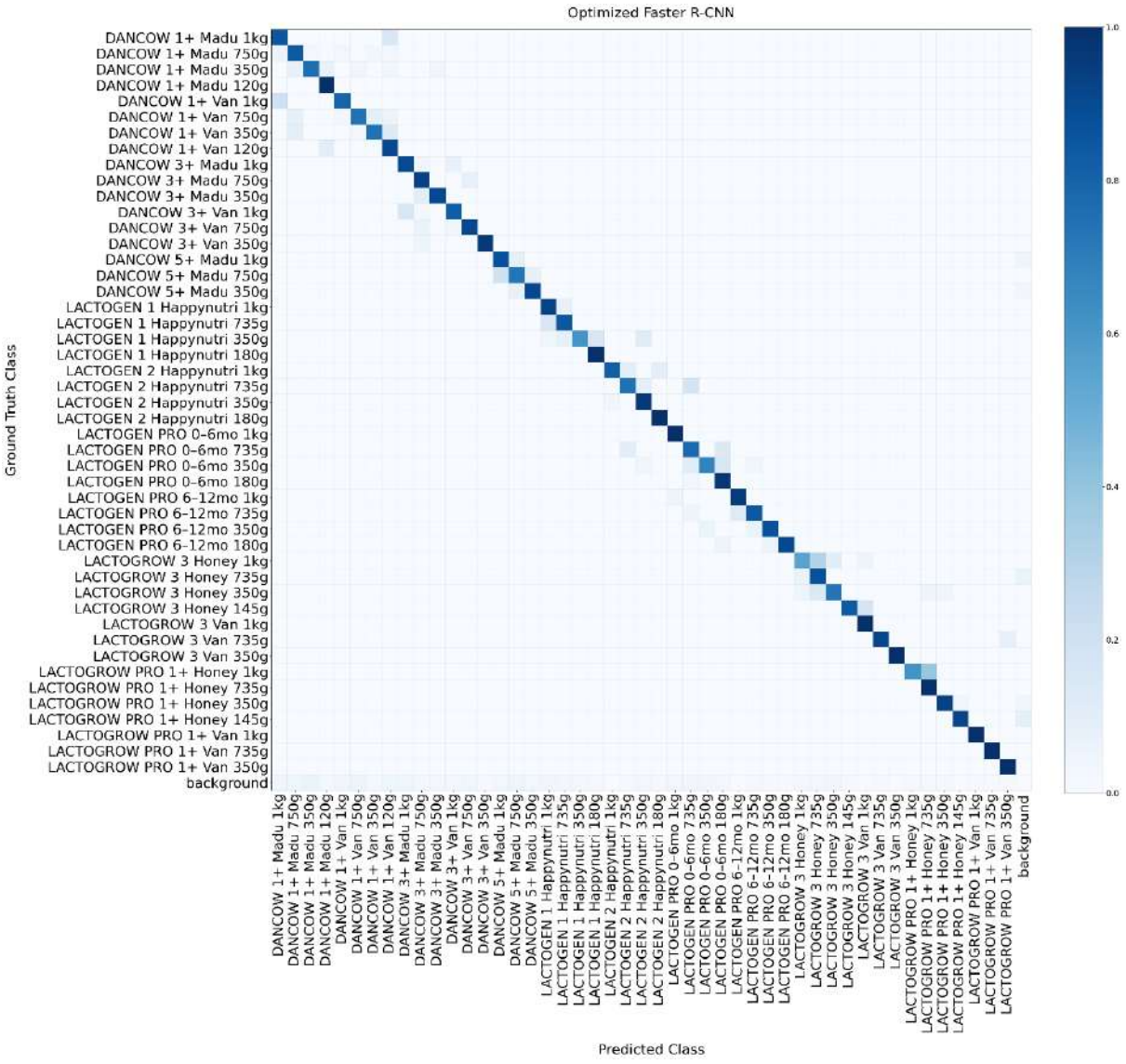


Figure 4.15 Optimized Faster R-CNN r50v2 confusion matrix

A detailed misclassification breakdown is compiled to examine class-level detection outcomes under the optimized configuration. The breakdown reports the dominant confusion patterns for each product variant. The optimized Faster R-CNN misclassification statistics are summarized in Table 4.17.

Table 4.17 Optimized Faster R-CNN r50v2 misclassification breakdown

ID	Class / Product Name	GT	TP	FN / Misclassified by
0	DANCOW 1+ Madu 1kg	20	17	• DANCOW 1+ Van 120g (3)
1	DANCOW 1+ Madu 750g	37	31	• DANCOW 1+ Madu 1kg (2) • DANCOW 1+ Madu 350g (1) • DANCOW 1+ Van 1kg (1) • DANCOW 1+ Van 350g (1) • DANCOW 1+ Van 120g (1)
2	DANCOW 1+ Madu 350g	29	22	• DANCOW 1+ Madu 750g (2) • DANCOW 1+ Madu 120g (2) • DANCOW 1+ Van 750g (1) • DANCOW 1+ Van 120g (1) • DANCOW 3+ Madu 350g (1)
3	DANCOW 1+ Madu 120g	48	48	
4	DANCOW 1+ Van 1kg	15	12	• DANCOW 1+ Madu 1kg (3)
5	DANCOW 1+ Van 750g	35	26	• DANCOW 1+ Madu 750g (3) • DANCOW 1+ Van 1kg (1) • DANCOW 1+ Van 350g (3) • DANCOW 1+ Van 120g (2)
6	DANCOW 1+ Van 350g	32	24	• DANCOW 1+ Madu 750g (3) • DANCOW 1+ Van 750g (1) • DANCOW 1+ Van 120g (4)
7	DANCOW 1+ Van 120g	20	18	• DANCOW 1+ Madu 120g (2)
8	DANCOW 3+ Madu 1kg	29	26	• DANCOW 3+ Madu 750g (1) • DANCOW 3+ Van 1kg (2)
9	DANCOW 3+ Madu 750g	28	26	• DANCOW 3+ Van 750g (2)
10	DANCOW 3+ Madu 350g	40	36	• DANCOW 3+ Madu 750g (4)
11	DANCOW 3+ Van 1kg	33	27	• DANCOW 3+ Madu 1kg (5) • DANCOW 3+ Madu 750g (1)
12	DANCOW 3+ Van 750g	31	28	• DANCOW 3+ Madu 750g (2) • DANCOW 3+ Van 1kg (1)
13	DANCOW 3+ Van 350g	49	47	• DANCOW 3+ Madu 750g (2)
14	DANCOW 5+ Madu 1kg	24	21	• DANCOW 5+ Madu 750g (2) • Background (1)
15	DANCOW 5+ Madu 750g	26	19	• DANCOW 5+ Madu 1kg (5) • DANCOW 5+ Madu 350g (2)
16	DANCOW 5+ Madu 350g	41	37	• DANCOW 5+ Madu 750g (3) • Background (1)
17	LACTOGEN 1 Happynutri 1kg	13	12	• LACTOGEN 1 Happynutri 735g (1)
18	LACTOGEN 1 Happynutri 735g	13	11	• LACTOGEN 1 Happynutri 1kg (2)
19	LACTOGEN 1 Happynutri 350g	26	16	• LACTOGEN 1 Happynutri 1kg (1) • LACTOGEN 1 Happynutri 735g (2) • LACTOGEN 1 Happynutri 180g (4) • LACTOGEN 2 Happynutri 350g (3)
20	LACTOGEN 1 Happynutri 180g	13	13	
21	LACTOGEN 2 Happynutri 1kg	11	9	• LACTOGEN 2 Happynutri 735g (1) • LACTOGEN 2 Happynutri 180g (1)

22	LACTOGEN 2 Happynutri 735g	12	9	• LACTOGEN 2 Happynutri 350g (1) • LACTOGEN PRO 0–6 mo 735g (2)
23	LACTOGEN 2 Happynutri 350g	26	25	• LACTOGEN 2 Happynutri 1kg (1)
24	LACTOGEN 2 Happynutri 180g	9	9	
25	LACTOGEN PRO 0-6 mo 1kg	12	12	
26	LACTOGEN PRO 0-6 mo 735g	31	24	• LACTOGEN 2 Happynutri 735g (3) • LACTOGEN PRO 0–6 mo 180g (4)
27	LACTOGEN PRO 0-6 mo 350g	28	19	• LACTOGEN 2 Happynutri 350g (1) • LACTOGEN PRO 0–6 mo 735g (3) • LACTOGEN PRO 0–6 mo 180g (4) • LACTOGEN PRO 6–12 mo 735g (1)
28	LACTOGEN PRO 0-6 mo 180g	33	32	• LACTOGEN PRO 0–6 mo 735g (1)
29	LACTOGEN PRO 6-12 mo 1kg	21	20	• LACTOGEN PRO 0–6 mo 1kg (1)
30	LACTOGEN PRO 6-12 mo 735g	25	21	• LACTOGEN PRO 0–6 mo 735g (1) • LACTOGEN PRO 6–12 mo 1kg (3)
31	LACTOGEN PRO 6-12 mo 350g	16	14	• LACTOGEN PRO 0–6 mo 350g (1) • LACTOGEN PRO 6–12 mo 735g (1)
32	LACTOGEN PRO 6-12 mo 180g	21	19	• LACTOGEN PRO 0–6 mo 180g (1) • LACTOGEN PRO 6–12 mo 350g (1)
33	LACTOGROW 3 Honey 1kg	20	11	• LACTOGROW 3 Honey 735g (6) • LACTOGROW 3 Honey 350g (2) • LACTOGROW 3 Van 1kg (1)
34	LACTOGROW 3 Honey 735g	16	14	• LACTOGROW 3 Honey 1kg (1) • Background (1)
35	LACTOGROW 3 Honey 350g	22	16	• LACTOGROW 3 Honey 1kg (1) • LACTOGROW 3 Honey 735g (3) • LACTOGROW PRO 1+ Honey 735g (1) • LACTOGROW PRO 1+ Honey 350g (1)
36	LACTOGROW 3 Honey 145g	6	5	• LACTOGROW 3 Van 1kg (1)
37	LACTOGROW 3 Van 1kg	12	12	
38	LACTOGROW 3 Van 735g	12	11	• LACTOGROW PRO 1+ Van 350g (1)
39	LACTOGROW 3 Van 350g	5	5	
40	LACTOGROW PRO 1+ Honey 1kg	13	8	• LACTOGROW PRO 1+ Honey 735g (5)
41	LACTOGROW PRO 1+ Honey 735g	8	8	
42	LACTOGROW PRO 1+ Honey 350g	28	26	• LACTOGROW PRO 1+ Honey 145g (1) • Background (1)
43	LACTOGROW PRO 1+ Honey 145g	12	11	• Background (1)
44	LACTOGROW PRO 1+ Van 1kg	11	11	
45	LACTOGROW PRO 1+ Van 735g	11	11	
46	LACTOGROW PRO 1+ Van 350g	12	12	

Despite the persistence of limited residual misclassification, the enhanced capability of the optimized Faster R-CNN in fine-grained detection is quantified by the per-class AP50:95 values presented in Table 4.18.

Table 4.18 Optimized Faster R-CNN r50v2 per-class average precision

ID	Class / Product Name	Average Precision (AP50:95)
0	DANCOW 1+ Madu 1kg	0.919019
1	DANCOW 1+ Madu 750g	0.941748
2	DANCOW 1+ Madu 350g	0.900585
3	DANCOW 1+ Madu 120g	0.932206
4	DANCOW 1+ Van 1kg	0.881100
5	DANCOW 1+ Van 750g	0.876673
6	DANCOW 1+ Van 350g	0.907858
7	DANCOW 1+ Van 120g	0.855359
8	DANCOW 3+ Madu 1kg	0.831021
9	DANCOW 3+ Madu 750g	0.854619
10	DANCOW 3+ Madu 350g	0.906736
11	DANCOW 3+ Van 1kg	0.829171
12	DANCOW 3+ Van 750g	0.805738
13	DANCOW 3+ Van 350g	0.903744
14	DANCOW 5+ Madu 1kg	0.866091
15	DANCOW 5+ Madu 750g	0.770067
16	DANCOW 5+ Madu 350g	0.900755
17	LACTOGEN 1 Happynutri 1kg	0.900540
18	LACTOGEN 1 Happynutri 735g	0.870196
19	LACTOGEN 1 Happynutri 350g	0.915743
20	LACTOGEN 1 Happynutri 180g	0.9135
21	LACTOGEN 2 Happynutri 1kg	0.896700
22	LACTOGEN 2 Happynutri 735g	0.812284
23	LACTOGEN 2 Happynutri 350g	0.884441
24	LACTOGEN 2 Happynutri 180g	0.940924
25	LACTOGEN PRO 0-6 mo 1kg	0.875501
26	LACTOGEN PRO 0-6 mo 735g	0.8242
27	LACTOGEN PRO 0-6 mo 350g	0.768374
28	LACTOGEN PRO 0-6 mo 180g	0.776056
29	LACTOGEN PRO 6-12 mo 1kg	0.778460
30	LACTOGEN PRO 6-12 mo 735g	0.817053
31	LACTOGEN PRO 6-12 mo 350g	0.826402
32	LACTOGEN PRO 6-12 mo 180g	0.834659
33	LACTOGROW 3 Honey 1kg	0.752768
34	LACTOGROW 3 Honey 735g	0.767331
35	LACTOGROW 3 Honey 350g	0.904303
36	LACTOGROW 3 Honey 145g	0.697648
37	LACTOGROW 3 Van 1kg	0.854632
38	LACTOGROW 3 Van 735g	0.879185
39	LACTOGROW 3 Van 350g	0.938944
40	LACTOGROW PRO 1+ Honey 1kg	0.822978
41	LACTOGROW PRO 1+ Honey 735g	0.912624
42	LACTOGROW PRO 1+ Honey 350g	0.819092

43	LACTOGROW PRO 1+ Honey 145g	0.813725
44	LACTOGROW PRO 1+ Van 1kg	0.923331
45	LACTOGROW PRO 1+ Van 735g	0.8979
46	LACTOGROW PRO 1+ Van 350g	0.867076

The optimized Faster R-CNN results provide a two-stage detection reference under the tuned configuration. In the next subsection, an ensemble strategy based on Weighted Boxes Fusion (WBF) is evaluated to examine whether complementary detection strengths from both YOLOv12x and Faster R-CNN can further improve detection accuracy.

4.3.5 WBF Ensemble of YOLOv12x and Faster R-CNN r50v2

This subsection evaluates the ensemble configuration that combines predictions from YOLOv12 and Faster R-CNN using Weighted Boxes Fusion (WBF). The confusion matrix is analyzed to examine changes in false-positive distribution relative to the standalone optimized detectors, while per-class AP50:95 values summarize detection performance across SKU variants under the fused prediction setting.

To evaluate the combined detection performance of YOLOv12x and Faster R-CNN, an ensemble configuration is constructed using Weighted Boxes Fusion (WBF). A normalized confusion matrix is generated to analyze the distribution of detection errors under the fused prediction setting. The resulting confusion matrix is shown in Figure 4.16.

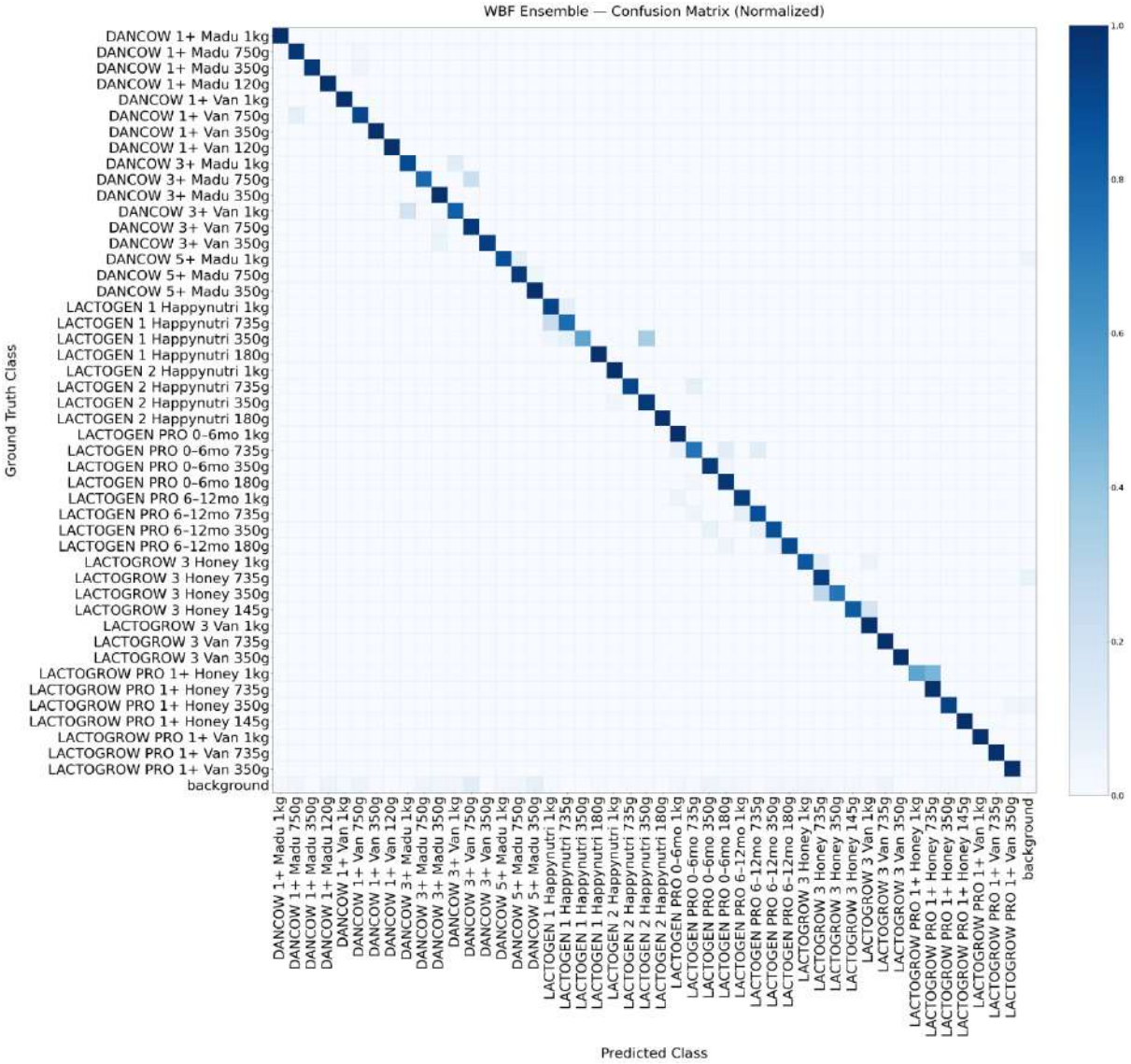


Figure 4.16 WBF Ensemble confusion matrix

A detailed misclassification breakdown is compiled for the WBF ensemble to examine class-level detection outcomes after prediction fusion. The misclassification statistics for all product variants are summarized in Table 4.19.

Table 4.19 WBF Ensemble misclassification breakdown

ID	Class / Product Name	GT	TP	FN / Misclassified by (Count)
0	DANCOW 1+ Madu 1kg	20	20	
1	DANCOW 1+ Madu 750g	37	36	• DANCOW 1+ Van 750g (1)
2	DANCOW 1+ Madu 350g	29	28	• DANCOW 1+ Van 750g (1)
3	DANCOW 1+ Madu 120g	48	48	
4	DANCOW 1+ Van 1kg	15	15	
5	DANCOW 1+ Van 750g	35	32	• DANCOW 1+ Madu 750g (3)
6	DANCOW 1+ Van 350g	32	32	

7	DANCOW 1+ Van 120g	20	20	
8	DANCOW 3+ Madu 1kg	29	26	• DANCOW 3+ Van 1kg (3)
9	DANCOW 3+ Madu 750g	28	22	• DANCOW 3+ Van 750g (6)
10	DANCOW 3+ Madu 350g	40	40	
11	DANCOW 3+ Van 1kg	33	27	• DANCOW 3+ Madu 1kg (6)
12	DANCOW 3+ Van 750g	31	30	• DANCOW 3+ Madu 350g (1)
13	DANCOW 3+ Van 350g	49	46	• DANCOW 3+ Madu 350g (3)
14	DANCOW 5+ Madu 1kg	24	21	• DANCOW 5+ Madu 750g (2) • Background (1)
15	DANCOW 5+ Madu 750g	26	25	• DANCOW 5+ Madu 350g (1)
16	DANCOW 5+ Madu 350g	41	41	
17	LACTOGEN 1 Happynutri 1kg	13	12	• LACTOGEN 1 Happynutri 735g (1)
18	LACTOGEN 1 Happynutri 735g	13	10	• LACTOGEN 1 Happynutri 1kg (3)
19	LACTOGEN 1 Happynutri 350g	26	14	• LACTOGEN 1 Happynutri 1kg (1) • LACTOGEN 1 Happynutri 735g (2) • LACTOGEN 2 Happynutri 350g (9)
20	LACTOGEN 1 Happynutri 180g	13	13	
21	LACTOGEN 2 Happynutri 1kg	11	11	
22	LACTOGEN 2 Happynutri 735g	12	11	• LACTOGEN PRO 0-6 mo 735g (1)
23	LACTOGEN 2 Happynutri 350g	26	25	• LACTOGEN 2 Happynutri 1kg (1)
24	LACTOGEN 2 Happynutri 180g	9	9	
25	LACTOGEN PRO 0-6 mo 1kg	12	12	
26	LACTOGEN PRO 0-6 mo 735g	31	23	• LACTOGEN PRO 0-6 mo 1kg (2) • LACTOGEN PRO 0-6 mo 180g (3) • LACTOGEN PRO 6-12 mo 735g (3)
27	LACTOGEN PRO 0-6 mo 350g	28	27	• LACTOGEN PRO 0-6 mo 180g (1)
28	LACTOGEN PRO 0-6 mo 180g	33	32	• LACTOGEN PRO 0-6 mo 735g (1)
29	LACTOGEN PRO 6-12 mo 1kg	21	20	• LACTOGEN PRO 0-6 mo 1kg (1)
30	LACTOGEN PRO 6-12 mo 735g	25	22	• LACTOGEN PRO 0-6 mo 735g (1) • LACTOGEN PRO 6-12 mo 1kg (2)
31	LACTOGEN PRO 6-12 mo 350g	16	14	• LACTOGEN PRO 0-6 mo 350g (1) • LACTOGEN PRO 6-12 mo 735g (1)
32	LACTOGEN PRO 6-12 mo 180g	21	19	• LACTOGEN PRO 0-6 mo 180g (1) • LACTOGEN PRO 6-12 mo 350g (1)
33	LACTOGROW 3 Honey 1kg	20	17	• LACTOGROW 3 Honey 735g (2) • LACTOGROW 3 Van 1kg (1)
34	LACTOGROW 3 Honey 735g	16	15	• Background (1)
35	LACTOGROW 3 Honey 350g	22	16	• LACTOGROW 3 Honey 735g (6)
36	LACTOGROW 3 Honey 145g	6	5	• LACTOGROW 3 Van 1kg (1)
37	LACTOGROW 3 Van 1kg	12	12	

38	LACTOGROW 3 Van 735g	12	12	
39	LACTOGROW 3 Van 350g	5	5	
40	LACTOGROW PRO 1+ Honey 1kg	13	7	• LACTOGROW PRO 1+ Honey 735g (6)
41	LACTOGROW PRO 1+ Honey 735g	8	8	
42	LACTOGROW PRO 1+ Honey 350g	28	26	• LACTOGROW PRO 1+ Van 350g (1) • Background (1)
43	LACTOGROW PRO 1+ Honey 145g	12	12	
44	LACTOGROW PRO 1+ Van 1kg	11	11	
45	LACTOGROW PRO 1+ Van 735g	11	11	
46	LACTOGROW PRO 1+ Van 350g	12	12	

The consolidated effect of prediction fusion on class-level accuracy is quantified by the resulting per-class AP50:95 values for the WBF ensemble configuration, detailed in Table 4.20.

Table 4.20 WBF Ensemble per-class average precision

ID	Class / Product Name	Average Precision (AP50:95)
0	DANCOW 1+ Madu 1kg	0.9552
1	DANCOW 1+ Madu 750g	0.968458
2	DANCOW 1+ Madu 350g	0.9362
3	DANCOW 1+ Madu 120g	0.940693
4	DANCOW 1+ Van 1kg	0.938369
5	DANCOW 1+ Van 750g	0.927941
6	DANCOW 1+ Van 350g	0.970622
7	DANCOW 1+ Van 120g	0.900887
8	DANCOW 3+ Madu 1kg	0.911956
9	DANCOW 3+ Madu 750g	0.904443
10	DANCOW 3+ Madu 350g	0.960801
11	DANCOW 3+ Van 1kg	0.851835
12	DANCOW 3+ Van 750g	0.884899
13	DANCOW 3+ Van 350g	0.930493
14	DANCOW 5+ Madu 1kg	0.881681
15	DANCOW 5+ Madu 750g	0.942797
16	DANCOW 5+ Madu 350g	0.960532
17	LACTOGEN 1 Happynutri 1kg	0.822600
18	LACTOGEN 1 Happynutri 735g	0.880211
19	LACTOGEN 1 Happynutri 350g	0.915923
20	LACTOGEN 1 Happynutri 180g	0.925228
21	LACTOGEN 2 Happynutri 1kg	0.929208
22	LACTOGEN 2 Happynutri 735g	0.929170
23	LACTOGEN 2 Happynutri 350g	0.890157
24	LACTOGEN 2 Happynutri 180g	0.934653
25	LACTOGEN PRO 0-6 mo 1kg	0.9098
26	LACTOGEN PRO 0-6 mo 735g	0.886657

27	LACTOGEN PRO 0-6 mo 350g	0.942999
28	LACTOGEN PRO 0-6 mo 180g	0.910490
29	LACTOGEN PRO 6-12 mo 1kg	0.858430
30	LACTOGEN PRO 6-12 mo 735g	0.842115
31	LACTOGEN PRO 6-12 mo 350g	0.807569
32	LACTOGEN PRO 6-12 mo 180g	0.839305
33	LACTOGROW 3 Honey 1kg	0.791431
34	LACTOGROW 3 Honey 735g	0.905680
35	LACTOGROW 3 Honey 350g	0.945928
36	LACTOGROW 3 Honey 145g	0.809054
37	LACTOGROW 3 Van 1kg	0.963168
38	LACTOGROW 3 Van 735g	0.936464
39	LACTOGROW 3 Van 350g	0.972277
40	LACTOGROW PRO 1+ Honey 1kg	0.835675
41	LACTOGROW PRO 1+ Honey 735g	0.923515
42	LACTOGROW PRO 1+ Honey 350g	0.873996
43	LACTOGROW PRO 1+ Honey 145g	0.890317
44	LACTOGROW PRO 1+ Van 1kg	0.954185
45	LACTOGROW PRO 1+ Van 735g	0.902098
46	LACTOGROW PRO 1+ Van 350g	0.828557

4.4 Discussions

This subsection provides an overall assessment of the experimental results by synthesizing quantitative performance evaluation, qualitative visual analysis, and the practical limitations identified throughout the study. The discussion compares the applied detection approaches across baseline models, optimized standalone models, and the final ensemble configuration using Weighted Boxes Fusion (WBF). Quantitative assessment focuses on global detection performance as measured by mAP50:95 and False Negative Rate (FNR), enabling an objective comparison of detection accuracy and missed detections across configurations. In addition, qualitative case studies are presented to visually illustrate the behavioral differences among the five experimental setups in realistic retail shelf scenarios. Finally, the limitations derived from the conducted experiments are discussed to contextualize the findings and outline directions for future work.

4.4.1 Performance Assessment

A. Baseline and Optimized YOLOv12x

The error analysis reveals a substantial reduction in intra-family grammage confusion after optimization. In the baseline YOLOv12x model, severe misclassification is concentrated within specific product families, particularly LACTOGROW PRO 1+ Honey, LACTOGEN

PRO 0–6mo, and LACTOGROW 3 Honey, which exhibit the highest cumulative intra-family FP magnitudes. Dominant errors largely occur between adjacent grammage sizes (e.g., 1kg ↔ 735g, 735g ↔ 350g), indicating difficulty in distinguishing subtle packaging variations with highly similar layouts and color schemes.

After optimization, the overall intra-family FP magnitude decreases markedly across nearly all product families. Several high-error families in the baseline such as LACTOGEN PRO 0–6mo and DANCOW 1+ variants show a pronounced reduction in grammage misclassification. The remaining FP patterns in the optimized model are fewer, more localized, and predominantly limited to a small subset of visually challenging families, most notably LACTOGROW 3 Honey and LACTOGROW PRO 1+ Honey. This shift indicates that optimization effectively suppresses widespread variant-level confusion while preserving detection sensitivity.

A limited trade-off is observed in the form of slightly increased background FP contributions for certain families, such as LACTOGROW 3 Honey and DANCOW 3+ Van. This behavior indicates a shift in the dominant error mode from incorrect intra-family grammage classification toward low-magnitude background activations, rather than a degradation in variant-level discrimination. These background-related FP increases are relatively low in magnitude and do not offset the significant reduction in intra-family misclassification errors.

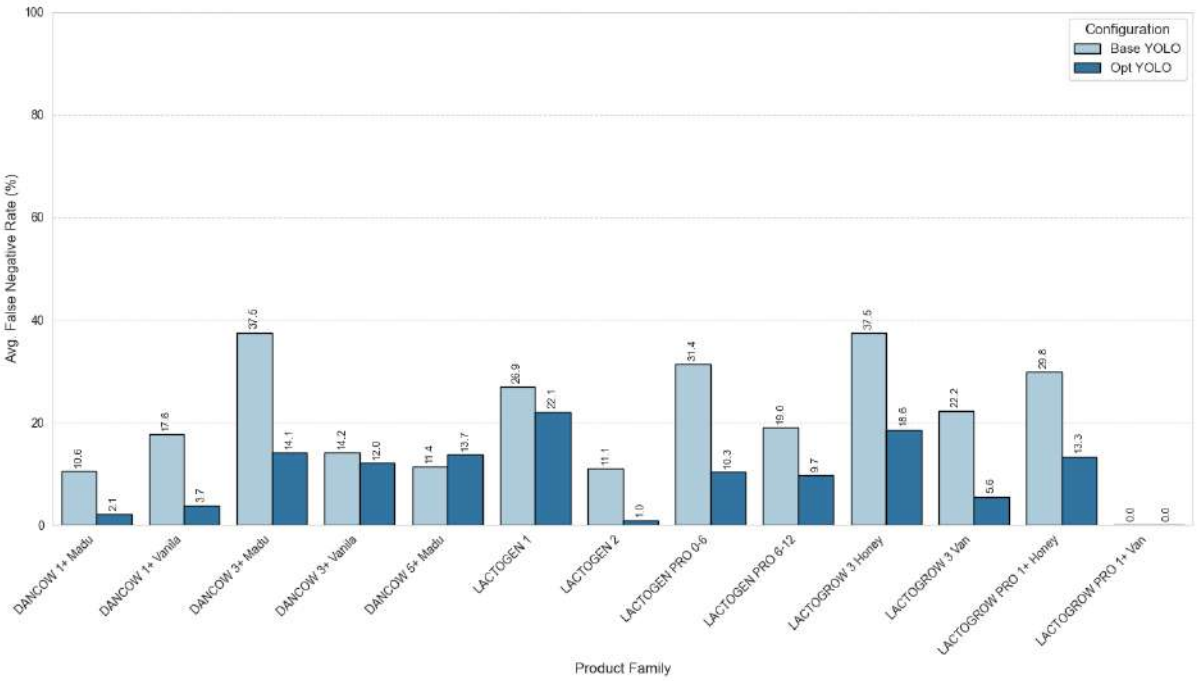


Figure 4.17 Baseline and optimized YOLOv12x false negative rate

Visually, Figure 4.17 illustrates a reduction in the False Negative Rate (FNR) bars across most product families, indicating that the optimization contributed to recovering previously missed instances. While specific values vary by family in the chart, the global average FNR calculated across the entire dataset further quantifies this improvement. The baseline model recorded a global average FNR of 20.71%, whereas the optimized configuration recorded 9.70%, a difference of 11.01 percentage points, reflecting a decrease in missed detections.

From a per-class performance perspective, the optimized YOLOv12x model demonstrates a more stable and balanced AP distribution across product families. Many previously underperforming variants, especially mid-range grammage sizes within visually similar families show noticeable AP improvements, reflecting enhanced fine-grained feature discrimination. High-performing classes in the baseline model maintain strong performance after optimization, while several weak classes experience meaningful gains.

Table 4.21 Baseline and optimized YOLOv12x mean average precision

Model	mAP50:95
Baseline YOLOv12	0.762
Optimized YOLOv12	**0.861**

At the global level, optimization yields a clear improvement in overall detection performance. The mAP50:95 increases from 0.762 (baseline) to 0.861 (optimized), corresponding to an absolute gain of +0.099.

Overall, the optimized YOLOv12x model demonstrates a clear and consistent improvement in fine-grained retail product detection. Optimization substantially suppresses intra-family grammage misclassification, the primary challenge targeted in this study, while delivering a significant global accuracy gain. Although minor background FP increases and isolated AP regressions persist, their impact is limited and outweighed by the broad reduction in variant-level confusion. These results indicate that the applied optimization strategy effectively enhances both robustness and discriminative capability for distinguishing visually similar SKU variants.

B. Baseline and Optimized Faster R-CNN r50v2

The error analysis indicates a substantial reduction in intra-family grammage confusion after optimization. In the baseline Faster R-CNN r50v2 model, severe misclassification is concentrated within several product families, most notably LACTOGEN PRO 0–6mo, LACTOGROW PRO 1+ Honey, and LACTOGEN PRO 6–12mo, which exhibit the highest cumulative intra-family FP magnitudes. Dominant errors primarily involve adjacent grammage sizes (e.g., 1kg ↔ 735g and 735g ↔ 350g), reflecting difficulty in separating highly similar packaging variants with minimal visual differences.

After optimization, the magnitude of intra-family FP decreases markedly across nearly all families. High-error families in the baseline such as LACTOGEN PRO 0–6mo, DANCOW 1+ Madu, and DANCOW 1+ Van show clear reductions in grammage confusion. The remaining FP patterns in the optimized model are fewer, more localized, and largely confined to visually challenging families, particularly LACTOGROW 3 Honey and LACTOGROW PRO 1+ Honey. This change indicates that optimization effectively suppresses widespread variant-level misclassification while retaining sensitivity to fine-grained visual cues.

In contrast to YOLOv12, the optimized Faster R-CNN exhibits a modest increase in background FP for a small number of families. Background FP contributions rise slightly for LACTOGROW PRO 1+ Honey and DANCOW 5+ Madu, suggesting increased responsiveness in cluttered shelf scenes. However, these increases remain small in magnitude and do not outweigh the pronounced reduction in intra-family grammage errors.

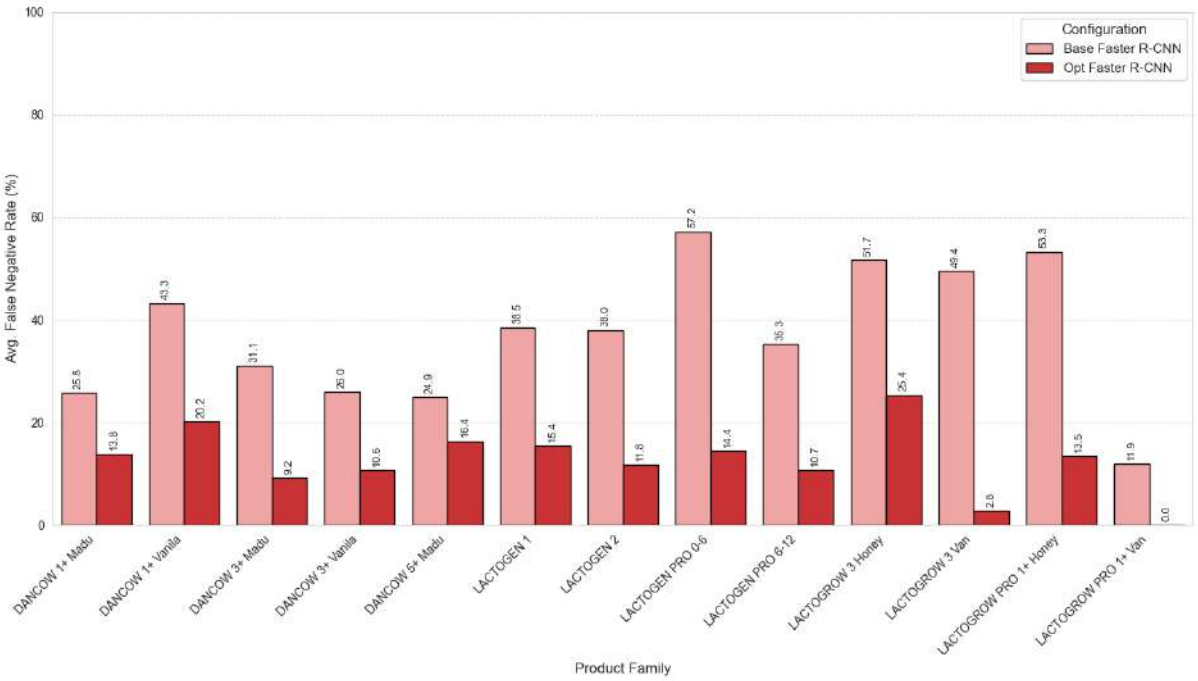


Figure 4.18 Baseline and optimized Faster R-CNN r50v2 false negative rate

Figure 4.18 presents the FNR magnitude for each product family in comparison to the baseline. Based on the statistical analysis, the global average FNR decreased from 37.40% in the baseline model to 12.62% in the optimized version. This difference of 24.77 percentage points suggests that the optimization process reduced the model's tendency to overlook or misclassify fine-grained variants.

The per-class AP results show broad and consistent improvements across most product families after optimization. Several previously low-performing variants experience substantial AP gains, particularly within families that exhibited strong grammage confusion in the baseline model. These improvements indicate enhanced fine-grained feature extraction and stronger discrimination between visually similar variants.

Table 4.22 Baseline and optimized Faster R-CNN r50v2 mean average precision

Model	mAP50:95
Baseline Faster R-CNN	0.709
Optimized Faster R-CNN	**0.859**

A small subset of classes exhibits modest AP regressions after optimization, primarily within visually subtle or low-instance categories. However, these declines are limited in scope and magnitude and do not alter the overall performance trend. At the global level, the optimized

Faster R-CNN model achieves a clear improvement in detection accuracy, with mAP50:95 increasing from 0.709 to 0.859, corresponding to an absolute gain of +0.15.

While optimization increases Faster R-CNN's background sensitivity in dense retail environments, this trade-off is less detrimental than the baseline error mode, as it replaces frequent grammage misclassification with lower-impact background detections. This trade-off reflects a shift toward more aggressive feature utilization, which reduces variant-level misclassification at the cost of marginally higher background detections. Importantly, the dominant error mode transitions from incorrect grammage prediction to low-magnitude background FP, which is less detrimental for downstream analysis.

Overall, optimization substantially improves Faster R-CNN's performance for fine-grained retail product detection. The model exhibits a pronounced reduction in intra-family grammage confusion, the primary challenge addressed in this study, alongside a significant global accuracy increase. Although minor background FP increases and isolated AP regressions persist, their impact is limited and outweighed by the gains in variant-level separability. These results confirm that the optimized Faster R-CNN model develops a more reliable and robust representation for distinguishing visually similar SKU variants compared with its baseline counterpart.

C. Standalone Optimized Models and WBF Ensemble

The error analysis of the WBF ensemble indicates that model fusion tends to reduce intra-family grammage confusion compared with the standalone detectors, while retaining only a small number of residual errors. The highest cumulative intra-family FP magnitudes in the ensemble are concentrated within LACTOGROW PRO 1+ Honey, LACTOGEN 1, and LACTOGROW 3 Honey families. These families are known to exhibit highly similar packaging layouts across grammages, making them inherently challenging even after fusion.

Compared to the standalone optimized models, dominant intra-family misclassification patterns are reduced and significantly more localized. The remaining FP transitions are largely restricted to adjacent grammages, such as 1kg ↔ 735g and 735g ↔ 350g, with substantially lower magnitudes. This behavior suggests that WBF consolidates complementary predictions from YOLOv12 and Faster R-CNN, reducing widespread variant-level confusion while preserving fine-grained sensitivity.

Importantly, the ensemble does not amplify background-related false positives beyond those observed in the optimized standalone models. These values remain relatively small, confirming that fusion consolidates predictions without introducing excessive over-detection.

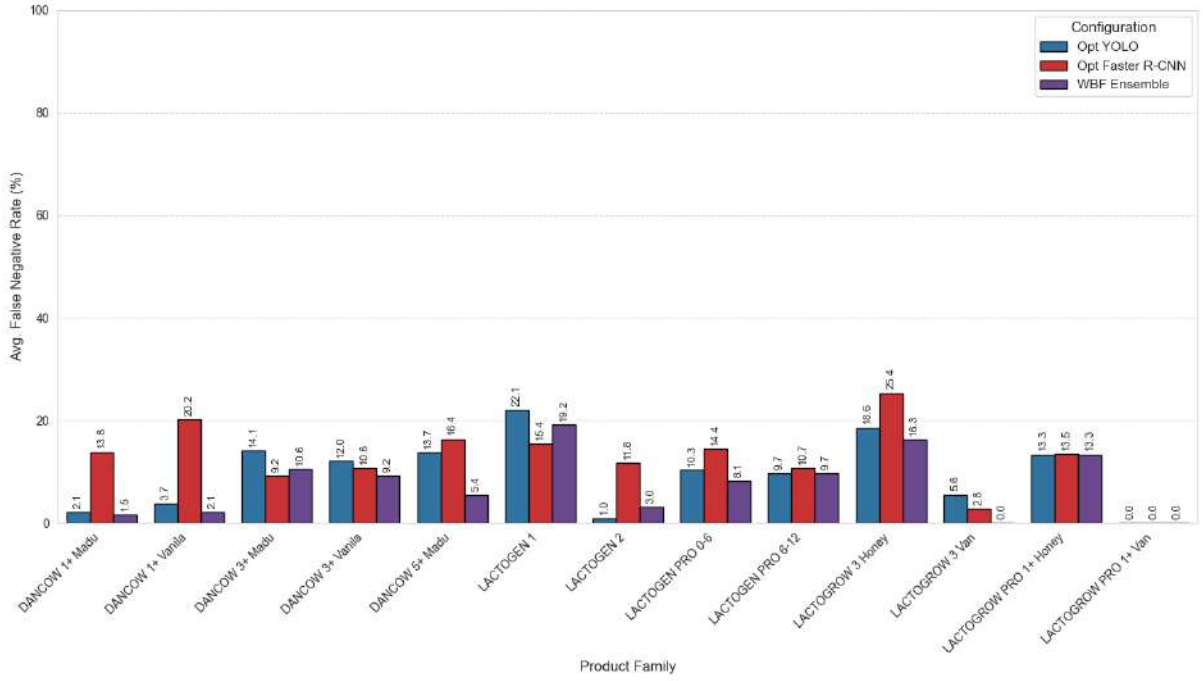


Figure 4.19 Optimized models and WBF ensemble false negative rate

As observed in Figure 4.19, the ensemble strategy exhibits the lowest error profile among the evaluated configurations. While specific product families show variation, the WBF ensemble achieved the most consistent performance overall. Quantitatively, the WBF ensemble recorded a global average FNR of 7.58%, which is lower than both the optimized YOLOv12 (9.70%) and Faster R-CNN (12.62%). This indicates that fusing predictions provided a net gain in overall reliability by recovering valid detections that were missed by the individual models.

From a performance perspective, the ensemble configuration yields a more uniform and stable AP distribution across product families. Many variants achieve their highest AP values under WBF, particularly within families that previously exhibited residual ambiguity in the standalone models. This improvement reflects the ensemble's ability to leverage complementary strengths of YOLOv12 and Faster R-CNN.

Table 4.23 Optimized models and WBF ensemble mean average precision

Model	mAP50:95
Optimized YOLOv12	0.861
Optimized Faster R-CNN	0.859
Ensemble of YOLOv12 and Faster R-CNN	**0.905**

From a global performance perspective, the WBF ensemble achieves a clear improvement over both optimized standalone models. Compared with optimized YOLOv12, the ensemble increases +0.044, while relative to optimized Faster R-CNN, the improvement is +0.046. These gains suggest that the ensemble combines complementary detection characteristics from both architectures, resulting in higher overall detection accuracy compared with the standalone optimized models. As a result, WBF delivers a higher and more stable overall detection accuracy than either optimized model individually.

Overall, the WBF ensemble provides an additional performance refinement rather than a radical behavioral change. While it does not fully eliminate grammage confusion in visually near-identical SKUs, it consistently reduces the severity and spread of such errors. These results confirm that ensemble fusion is effective as a final enhancement stage, producing more balanced fine-grained predictions with limited additional trade-offs, and complementing the optimized standalone detectors in challenging retail shelf environments.

Taken together, these findings demonstrate that the proposed optimization pipeline, comprising model-level tuning and final ensemble fusion, systematically improves fine-grained SKU discrimination while maintaining stable global detection performance.

D. Overall Performance of All Experiments

To provide an overview of the experimental progression, Table 4.24 and Table 4.25 summarize the performance metrics across all five evaluated configurations: Baseline YOLOv12, Optimized YOLOv12, Baseline Faster R-CNN, Optimized Faster R-CNN, and the final WBF Ensemble.

These tables highlight the incremental improvements achieved at each stage of the pipeline. The optimization process significantly reduced the Global Average False Negative Rate (FNR) for both architectures, recovering missed detections caused by fine-grained confusion. Subsequently, the WBF Ensemble further minimized these errors, achieving the lowest FNR of 7.58% and the highest mean Average Precision (mAP) of 0.905.

Table 4.24 Comparison of FNR across all experimental configurations

Model	FNR
Baseline YOLOv12	20.71%
Baseline Faster R-CNN	37.40%
Optimized YOLOv12	9.70%
Optimized Faster R-CNN	12.62%
Ensemble of YOLOv12 and Faster R-CNN	**7.58%**

Table 4.25 Comparison of mAP50:95 across all experimental configurations

Model	mAP50:95
Baseline YOLOv12	0.762
Baseline Faster R-CNN	0709
Optimized YOLOv12	0.861
Optimized Faster R-CNN	0.859
Ensemble of YOLOv12 and Faster R-CNN	**0.905**

A visual comparison of the confusion matrices across all experimental configurations is presented in Figure 4.20. This figure provides a birdview of the previous confusion matrices.

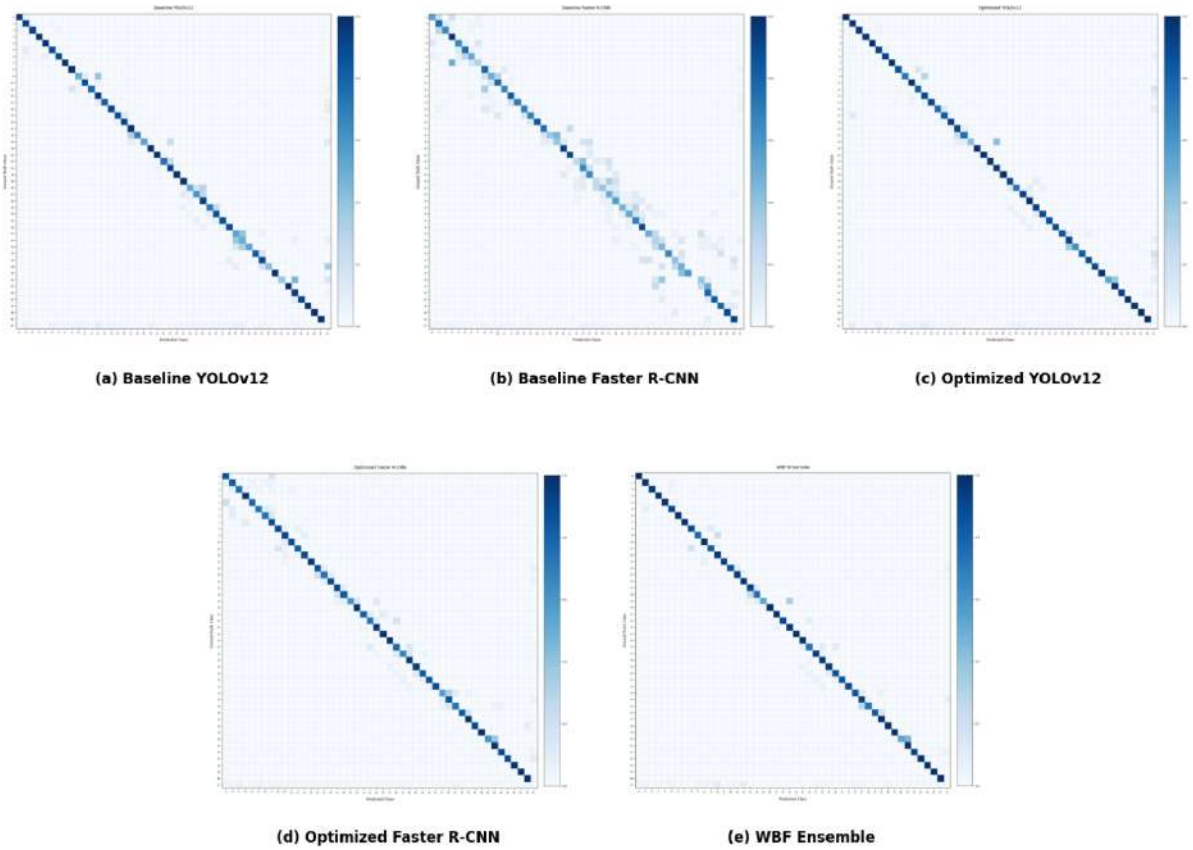


Figure 4.20 Comparison of confusion matrices across all model experiments

4.4.2 Visual Assessment on Shelf Scenarios

A. Intra-Family Class Variation

This case evaluates the model's robustness against significant scale variations within a single product category. As shown in Figure 4.21, the scene features three DANCOW 1+ Madu packs of distinctly different physical sizes (1kg, 750g, and 350g) placed side-by-side, presenting a challenge in recognizing identical visual features across varying spatial resolutions.

Figure 4.21 Case 1 ground truth

The detection performance across the five experimental configurations, ranging from the baselines to the final WBF ensemble is presented in Figure 4.22. This comparison visually demonstrates how the models perform in detecting representative product with size variants.

(a) Baseline YOLOv12

(b) Baseline Faster R-CNN

(c) Optimized YOLOv12

(d) Optimized Faster R-CNN

(e) WBF

Figure 4.22 Case 1 visual outcomes of five experiments

To complement the qualitative visual comparison shown in Figure 4.21, the detection outcomes for Case 1 are summarized in a structured form in Table 4.26. The table reports the prediction results of each experimental configuration, detailing true positive detections along with their corresponding confidence scores for the size-variant instances observed in the scene.

Table 4.26 Case 1 experimental results

Ground Truth (Count)	Model	Prediction (Confidence)
3 total instances: • DANCOW 1+ Madu 1kg (1) • DANCOW 1+ Madu 750g (1) • DANCOW 1+ Madu 350g (1)	Baseline YOLOv12	TP: • DANCOW 1+ Madu 1kg (0.983) • DANCOW 1+ Madu 750g (0.987) • DANCOW 1+ Madu 350g (0.926) (TP=3, FP=0, Missing=0)
	Baseline Faster R-CNN	TP: • DANCOW 1+ Madu 1kg (0.680) • DANCOW 1+ Madu 750g (0.929) • DANCOW 1+ Madu 350g (0.946) (TP=3, FP=0, Missing=0)
	Optimized YOLOv12	TP: • DANCOW 1+ Madu 1kg (0.927) • DANCOW 1+ Madu 750g (0.971) • DANCOW 1+ Madu 350g (0.963) (TP=3, FP=0, Missing=0)
	Optimized Faster R-CNN	TP: • DANCOW 1+ Madu 1kg (0.998) • DANCOW 1+ Madu 750g (0.999) • DANCOW 1+ Madu 350g (0.997) (TP=3, FP=0, Missing=0)
	WBF Ensemble	TP: • DANCOW 1+ Madu 1kg (0.951) • DANCOW 1+ Madu 750g (0.980) • DANCOW 1+ Madu 350g (0.974) (TP=3, FP=0, Missing=0)

B. Case 2: Multiple Product Families

Case 2 examines model performance in a mixed-family shelf scenario involving multiple closely related product lines and grammage variants within a single frame. As shown in Figure 4.23, the ground truth includes a combination of LACTOGEN 1 Happynutri, multiple grammage variants of LACTOGEN PRO 0–6 months, and a smaller presence of LACTOGROW 3 Honey.

Figure 4.23 Case 2 ground truth

Figure 4.24 illustrates the results from the five experimental setups. By comparing the baseline outputs with the optimized and ensemble results, the progression in distinguishing these closely related classes and the improvement in confidence scores become evident.

Figure 4.24 Case 2 visual output of five experiments

Following the visual inspection of the five experimental outputs in Figure 4.24, a quantitative breakdown of the detection results is provided in Table 4.27. This table consolidates the prediction outcomes across all evaluated models, including true positives, false positives, and missed detections, enabling a clearer comparison of model behavior under a mixed-family shelf scenario.

Table 4.27 Case 2 experimental results

Ground Truth (Count)	**Model**	**Prediction (Confidence)**
8 total instances: • LACTOGEN PRO 0-6mo 735g (1) • LACTOGEN PRO 0-6mo 350g (2) • LACTOGEN PRO 0-6mo 180g (2) • LACTOGEN 1 Happynutri 735g (1) • LACTOGEN 1 Happynutri 350g (1) • LACTOGROW 3 Honey 350g (1)	Baseline YOLOv12	TP: • LACTOGEN PRO 0-6mo 180g (0.937) FP: • LACTOGEN PRO 0-6mo 1kg (0.572) • LACTOGEN PRO 0-6mo 180g (0.976 & 0.926) • LACTOGEN 1 Happynutri 1kg (0.974) • LACTOGROW 3 Honey 735g (0.864) Missing: • LACTOGEN PRO 0-6mo 735g • LACTOGEN PRO 0-6mo 350g • LACTOGEN PRO 0-6mo 180g • LACTOGEN 1 Happynutri 735g • LACTOGEN 1 Happynutri 350g • LACTOGROW 3 Honey 350g (TP=1, FP=5, Missing=7)
	Baseline Faster R-CNN	TP: • LACTOGEN PRO 0-6mo 735g (0.618) • LACTOGEN 1 Happynutri 735g (0.974) FP: • LACTOGEN PRO 6-12mo 1kg (0.544) • LACTOGEN PRO 6-12mo 350g (0.706, 0.973, and 0.926) • LACTOGEN 1 Happynutri 735g (0.747) • LACTOGROW PRO 1+ Honey 735g (0.634) Missing: • LACTOGEN PRO 0-6mo 350g • LACTOGEN PRO 0-6mo 180g • LACTOGEN 1 Happynutri 350g • LACTOGROW 3 Honey 350g (TP=2, FP=6, Missing=6)

	Optimized YOLOv12	TP: • LACTOGEN PRO 0-6mo 735g (0.954) • LACTOGEN PRO 0-6mo 350g (0.930 & 0.976) • LACTOGEN PRO 0-6mo 180g (0.643 & 0.957) • LACTOGEN 1 Happynutri 350g (0.567) FP: • LACTOGEN PRO 0-6mo 735g (0.597) • LACTOGEN 1 Happynutri 1kg (0.914) • LACTOGROW 3 Honey 735g (0.526) Missing: • LACTOGEN 1 Happynutri 735g • LACTOGROW 3 Honey 350g (TP=6, FP=3, Missing=2)
	Optimized Faster R-CNN	TP: • LACTOGEN PRO 0-6mo 735g (0.999) • LACTOGEN PRO 0-6mo 350g (0.648 & 0.925) • LACTOGEN PRO 0-6mo 180g (0.993 & 0.998) • LACTOGEN 1 Happynutri 735g (0.997) • LACTOGROW 3 Honey 350g (0.997) FP: • LACTOGEN PRO 0-6mo 735g (0.756) • LACTOGEN 1 Happynutri 1kg (0.984) • LACTOGEN PRO 6-12mo 350g (0.958) Missing: • LACTOGEN 1 Happynutri 735g (TP=7, FP=3, Missing=1)
	WBF Ensemble	TP: • LACTOGEN PRO 0-6mo 735g (0.977) • LACTOGEN PRO 0-6mo 350g (0.787 & 0.958) • LACTOGEN PRO 0-6mo 180g (0.891 & 0.972) • LACTOGEN 1 Happynutri 735g (0.898) • LACTOGROW 3 Honey 350g (0.602) FP: • LACTOGEN 1 Happynutri 1kg (0.853) Missing: • LACTOGEN 1 Happynutri 735g (TP=7, FP=1, Missing=1)

C. Case 3: High-Density Shelf Display

The final case investigates model performance in a complex, high-density retail environment characterized by significant occlusion. Figure 4.25 shows the Ground Truth, which contains tightly packed DANCOW variants (1+, 3+, and 5+) arranged in multiple layers. This scenario serves as a test for localization accuracy in cluttered scenes where bounding boxes frequently overlap.

Figure 4.25 Case 3 ground truth

The corresponding inference results are detailed in Figure 4.26. This comparison highlights the effectiveness of the proposed pipeline in separating adjacent objects and handling inter-object occlusion.

(a) Baseline YOLOv12

(b) Baseline Faster R-CNN

(c) Optimized YOLOv12

(d) Optimized Faster R-CNN

(e) WBF

Figure 4.26 Case 3 visual output of five experiments

To support the visual observations presented in Figure 4.26, the detailed detection results for the high-density shelf scenario are tabulated in Table 4.28. The table enumerates the prediction performance of each model configuration, highlighting detection accuracy, residual false positives, and missing instances in a cluttered and occluded environment.

Table 4.28 Case 3 experimental results

Ground Truth (Count)	**Model**	**Prediction (Confidence)**
16 total instances: • DANCOW 1+ Madu 1kg (1) • DANCOW 1+ Madu 750g (1) • DANCOW 1+ Van 1kg (1) • DANCOW 1+ Van 750g (2) • DANCOW 3+ Madu 1kg (1) • DANCOW 3+ Madu 750g (1) • DANCOW 3+ Madu 350g (1)	Baseline YOLOv12	TP: • DANCOW 1+ Madu 1kg (0.972) • DANCOW 1+ Madu 750g (0.972 & 0.965) • DANCOW 1+ Van 1kg (0.971) • DANCOW 1+ Van 750g (0.965 & 0.972) • DANCOW 3+ Madu 1kg (0.943) • DANCOW 3+ Madu 750g (0.750) • DANCOW 3+ Van 1kg (0.974 & 0.974) • DANCOW 3+ Van 750g (0.947 & 0.965) • DANCOW 5+ Madu 1kg (0.987 & 0.989) • DANCOW 5+ Madu 750g (0.861) FP: • DANCOW 3+ Madu 750g (0.720) • DANCOW 5+ Madu 750g (0.610) Missing: • DANCOW 3+ Madu 350g • DANCOW 3+ Van 350g (TP=14, FP=2, Missing=2)

• DANCOW 3+ Van 1kg (2) • DANCOW 3+ Van 750g (2) • DANCOW 3+ Van 350g (1) • DANCOW 5+ Madu 1kg (2) • DANCOW 5+ Madu 750g (1)	Baseline Faster R-CNN	TP: • DANCOW 1+ Madu 1kg (0.797) • DANCOW 1+ Madu 750g (0.960) • DANCOW 1+ Van 1kg (0.970) • DANCOW 1+ Van 750g (0.768 & 0.910) • DANCOW 3+ Madu 1kg (0.993) • DANCOW 3+ Madu 750g (0.964) • DANCOW 3+ Madu 350g (0.578) • DANCOW 3+ Van 1kg (0.981 & 0.985) • DANCOW 3+ Van 750g (0.544) • DANCOW 3+ Van 350g (0.849) • DANCOW 5+ Madu 1kg (0.677) • DANCOW 5+ Madu 750g (0.972) FP: • DANCOW 3+ Madu 1kg (0.739) • DANCOW 3+ Van 750g (0.937) • DANCOW 3+ Van 350g (0.748 & 0.945) • DANCOW 5+ Madu 350g (0.966) Missing: • DANCOW 3+ Van 750g • DANCOW 5+ Madu 1kg (TP=14, FP=5, Missing=2)
	Optimized YOLOv12	TP: • DANCOW 1+ Madu 1kg (0.958) • DANCOW 1+ Madu 750g (0.972) • DANCOW 1+ Van 1kg (0.985) • DANCOW 1+ Van 750g (0.967) • DANCOW 3+ Madu 1kg (0.925) • DANCOW 3+ Madu 750g (0.809) • DANCOW 3+ Madu 350g (0.945) • DANCOW 3+ Van 1kg (0.980) • DANCOW 3+ Van 750g (0.964) • DANCOW 3+ Van 350g (0.955) • DANCOW 5+ Madu 1kg (0.962) • DANCOW 5+ Madu 750g (0.915) FP: • DANCOW 5+ Madu 350g (0.967) Missing: • DANCOW 1+ Van 750g • DANCOW 3+ Van 1kg • DANCOW 3+ Van 750g • DANCOW 5+ Madu 1kg (TP=12, FP=1, Missing=4)

	Optimized Faster R-CNN	TP: • DANCOW 1+ Madu 1kg (0.999) • DANCOW 1+ Madu 750g (0.991) • DANCOW 1+ Van 1kg (0.999) • DANCOW 1+ Van 750g (0.977 & 0.992) • DANCOW 3+ Madu 1kg (0.997) • DANCOW 3+ Madu 750g (0.999) • DANCOW 3+ Madu 350g (0.989) • DANCOW 3+ Van 1kg (0.999 & 1.000) • DANCOW 3+ Van 750g (0.998 & 0.999) • DANCOW 3+ Van 350g (0.996) • DANCOW 5+ Madu 1kg (1.000 & 1.000) • DANCOW 5+ Madu 750g (0.984) FP: • DANCOW 3+ Van 750g (0.985) • DANCOW 5+ Madu 350g (0.988) (TP=16, FP=2, Missing=0)
	WBF Ensemble	TP: • DANCOW 1+ Madu 1kg (0.972) • DANCOW 1+ Madu 750g (0.979) • DANCOW 1+ Van 1kg (0.990) • DANCOW 1+ Van 750g (0.970 & 0.972) • DANCOW 3+ Madu 1kg (0.949) • DANCOW 3+ Madu 750g (0.872) • DANCOW 3+ Madu 350g (0.960) • DANCOW 3+ Van 1kg (0.981 & 0.986) • DANCOW 3+ Van 750g (0.972 & 0.976) • DANCOW 3+ Van 350g (0.968) • DANCOW 5+ Madu 1kg (0.971 & 0.974) • DANCOW 5+ Madu 750g (0.937) FP: • DANCOW 3+ Van 750g (0.328) • DANCOW 5+ Madu 350g (0.977) (TP=16, FP=2, Missing=0)

D. Qualitative Observations from Visual Assessments

This subsection provides a qualitative observation of the visual assessment results obtained from the evaluated shelf scenarios. The analysis focuses on observable model behavior, including detection consistency and characteristics across different levels of scene complexity. Based on the visual outputs discussed in the previous cases, the summarized observations for each scenario are presented in Table 4.29 to highlight the dominant qualitative patterns exhibited by the baseline models, optimized models, and ensemble model.

Table 4.29 Observations across visual assessment cases

Case	Observation
Case 1: Intra-Family Class Variation	All evaluated models successfully detected the three DANCOW 1+ Madu variants with no false positives or missed detections. The results indicate that scale variation within a single product family does not pose a significant challenge when the objects are captured at close range and exhibit strong visual cues. Confidence scores across both baseline and optimized models remain consistently high, suggesting stable feature recognition under low scene complexity.
Case 2: Multiple Product Families	Baseline models exhibit poor recall and frequent misclassification in mixed-family scenarios, indicating difficulty in distinguishing visually similar packaging across different product lines and age variants. Performance improves substantially after optimization, with a marked reduction in missing detections. The WBF ensemble further stabilizes detection by reducing false positives while maintaining high recall, demonstrating its effectiveness in resolving inter-family ambiguity under moderate scene complexity.
Case 3: High-Density Shelf Display	In a cluttered and occluded shelf environment, baseline YOLOv12 achieves higher recall than its optimized counterpart, indicating that a more permissive detection behavior can be advantageous in dense scenes. Optimized Faster R-CNN delivers the most reliable classification with no missed detections, highlighting its strength in fine-grained discrimination. The WBF ensemble matches this performance while maintaining consistent confidence across overlapping objects, confirming its robustness in high-density retail scenarios.

Overall, the visual assessment reveals consistent behavioral differences among the evaluated detection configurations. The baseline YOLOv12 model demonstrates a more selective detection behavior, producing a limited number of predictions that generally align with prominent object appearances. In contrast, the baseline Faster R-CNN exhibits higher sensitivity in region proposal generation, often producing a larger set of candidate detections, which increases object coverage but also introduces additional false positives, particularly in scenes with visually similar product variants.

The application of optimization strategies improves the overall stability of both architectures. Optimized YOLOv12 shows clearer decision boundaries and improved classification consistency in mixed-family scenarios, while optimized Faster R-CNN provides more reliable fine-grained classification under complex visual conditions. However, the results

also indicate that optimization does not guarantee superior performance in all cases. In highly cluttered scenes such as Case 3, the optimized YOLOv12 adopts a more conservative detection behavior, leading to reduced recall compared to its baseline counterpart. This finding highlights that models with stricter confidence thresholds may overlook valid objects when severe occlusion and overlap are present.

The benefit of model complementarity becomes most evident through the WBF ensemble, particularly in Case 3. Instances missed by YOLOv12 due to conservative detection are successfully recovered through Faster R-CNN predictions. This complementary interaction enables the ensemble to achieve full recall while maintaining controlled false positives. As a result, the WBF ensemble consistently delivers balanced and robust performance across all evaluated scenarios. Although confidence values may be moderated as an inherent effect of the fusion mechanism, the ensemble prioritizes correct class assignment and detection completeness. Given the objective of this study, which emphasizes accurate fine-grained product classification over raw confidence magnitude, the WBF ensemble is identified as the most optimal detection strategy among the evaluated methods.

4.4.3 Proposed End-to-End Fine-Grained Retail Detection Pipeline

A. Object Detection Model Selection

The proposed pipeline begins with the selection of representative object detection architectures. Rather than relying on a single detector, the framework adopts a dual-architecture strategy to exploit complementary detection characteristics. One-stage detectors are known for efficient global feature extraction, while two-stage detectors provide stronger region-level discrimination.

In this study, YOLOv12 and Faster R-CNN are chosen as the primary detection paradigms. Among the evaluated variants, YOLOv12x and Faster R-CNN with the ResNet-50-FPN-v2 backbone are selected as the baseline architectures due to their superior performance relative to other configurations within the same model families. These two models serve as the foundation for all subsequent optimization and inference stages in the proposed pipeline.

B. Performance-Driven Optimization and Dataset Enhancement

After establishing the baseline detection architectures, the proposed pipeline incorporates a performance-driven optimization stage to improve learning stability and fine-grained

discrimination capability. Hyperparameter optimization (HPO) is applied to both YOLOv12x and Faster R-CNN using an automated search strategy to refine learning-related parameters that influence convergence behavior, loss balancing, and feature sensitivity. In this study, the optimization process is conducted using the Optuna framework, where multiple hyperparameter configurations are evaluated and compared against the baseline performance. The configuration that yields the best evaluation performance is selected as the optimized setting for each architecture. This approach ensures that subsequent stages of the pipeline rely on empirically validated model configurations rather than default or manually tuned parameters.

In addition to model-level optimization, the pipeline integrates a dataset-level enhancement strategy to mitigate limitations caused by data scarcity and class imbalance. Context-Aware Copy-Paste augmentation is employed to enrich the training dataset by synthesizing additional samples for underrepresented product variants. This strategy places object instances into semantically compatible shelf environments, increasing data diversity while preserving realistic spatial arrangements.

Both detection architectures are retrained using the selected optimized hyperparameters and the augmented dataset. The resulting model weights represent the optimized detectors that are subsequently used for inference.

C. Model Inference

Inference in the proposed pipeline is performed in parallel using the optimized YOLOv12x and Faster R-CNN models. Each detector is configured with inference parameters that are deliberately adjusted to retain a sufficient number of candidate detections for ensemble fusion. These inference configurations are not chosen arbitrarily; instead, several parameter settings are evaluated through controlled experiments, and the configuration that produces the most reliable evaluation performance is selected.

Figure 4.27 shows the inference configuration used for YOLOv12x. The parameters are set to avoid premature confidence-based filtering while maintaining consistent image resolution and IoU settings.

```
results = model.predict(
    imgsz=640,
    conf=0.0,
```

```
    iou=0.5,
    max_det=500,
    rect=False,
    half=False,
    augment=False,
    agnostic_nms=False
)
```

Figure 4.27 Code snippet of YOLOv12x inference configuration

For Faster R-CNN, inference is configured to generate a large number of region proposals and retain overlapping bounding boxes. Similar to YOLOv12x, multiple inference configurations are evaluated, and the final parameter setting is selected based on its evaluation performance. Figure 4.28 presents the inference parameter configuration applied to Faster R-CNN.

```
results = model.predict(
    imgsz=640,
    conf=0.0,
    iou=0.5,
    max_det=500,
    rect=False,
    half=False,
    augment=False,
    agnostic_nms=False
)
```

Figure 4.28 Code snippet of Faster R-CNN r50v2 inference configuration

The inference outputs from both detectors, consisting of bounding box coordinates, confidence scores, and class labels, are retained without aggressive filtering. These outputs serve as the input for the ensemble fusion stage.

D. Ensemble Model

The final stage of the proposed pipeline applies an ensemble strategy using Weighted Boxes Fusion (WBF) to combine inference outputs from YOLOv12x and Faster R-CNN. The objective of this stage is to consolidate overlapping predictions from different detectors while preserving complementary detections that may be missed by a single model.

Several WBF parameter configurations are explored to find the best one that yields higher overall accuracy. The final configuration is selected based on its evaluation performance, ensuring that the fusion process contributes positively to the overall detection quality. Figure 4.29 shows the WBF configuration used in the proposed pipeline.

```
fused_boxes, fused_scores, fused_labels = weighted_boxes_fusion(
    boxes_list,
    scores_list,
    labels_list,
    weights=[2.0, 1.0],     # YOLOv12x, Faster R-CNN r50v2
    iou_thr=0.75,
    skip_box_thr=0.00001
)
```

Figure 4.29 Code snippet of WBF configuration

Following fusion, the resulting predictions are converted back to absolute image coordinates and form the final output of the proposed pipeline. The output consists of labeled retail shelf images with bounding boxes and class annotations for each detected product variant, representing the finalized detection results used for evaluation and visualization.

Figure 4.30 summarizes the diagram of overall end-to-end fine-grained retail detection pipeline proposed in this study. The diagram illustrates the proposed end-to-end fine-grained retail detection pipeline, detailing the flow from dataset preparation through performance-driven optimization and inference to the final detection output.

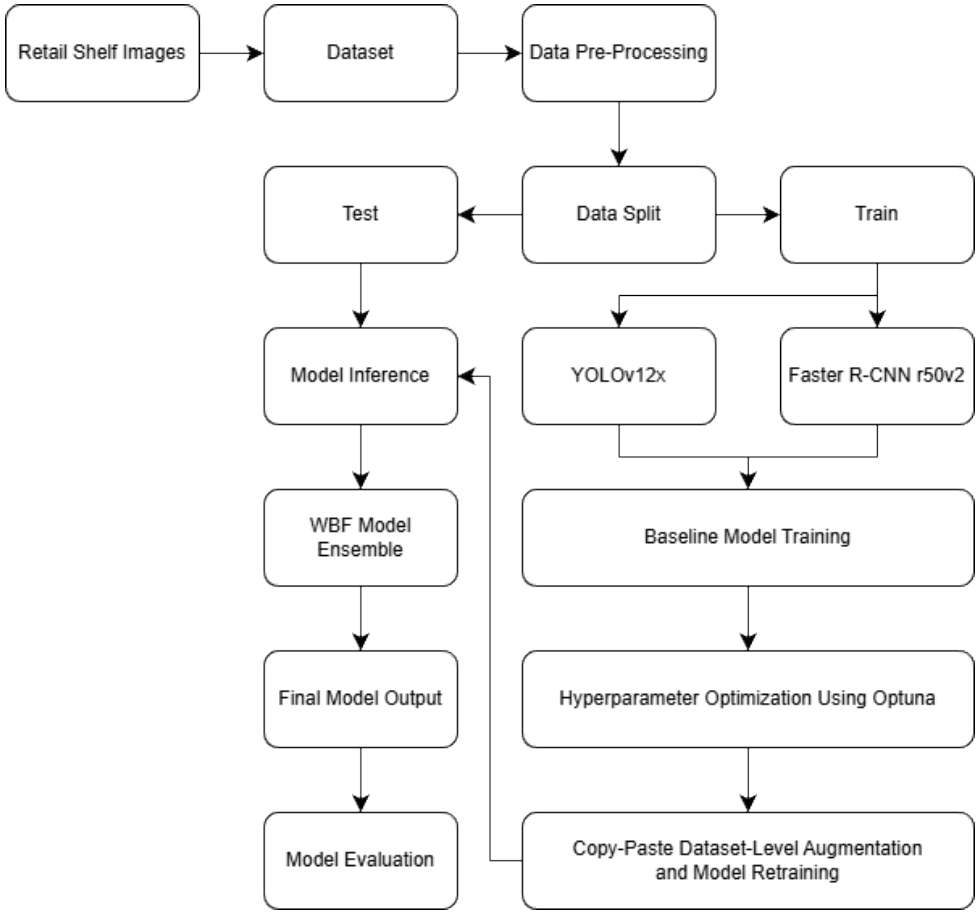


Figure 4.30 End-to-end retail product detection optimization pipeline

4.4.4 Limitations

This research encountered several constraints during the development process. These limitations are acknowledged to provide context for the results and to direct future improvements:

a. Data Constraints and Splitting Strategy: Due to the limited size of the curated fine-grained dataset, the study employed a two-subset split (Train and Test) rather than the standard Train-Validation-Test protocol. While this approach was necessary to maximize the training data volume under scarcity conditions, it may introduce a degree of bias during the hyperparameter optimization process as the test set was utilized for evaluation feedback.

b. Environmental Variability: The dataset predominantly consists of images captured under relatively controlled lighting conditions. The model's robustness against extreme real-world retail anomalies, such as severe lighting fluctuations, heavy glare on plastic

wrapping, or extreme occlusion in highly disordered clearance shelves remains to be fully verified in diverse physical deployments.

c. Limited Domain Generalization: The dataset constructed for this research is strictly confined to specific families of powdered milk products, all presented in similar box packaging. Consequently, the model's generalization capability across a broader spectrum of retail packaging form factors, such as bottles, cans, or transparent wrappers, has not been evaluated, limiting the verified scope of the proposed framework to box-packaged goods.

d. Restricted HPC Software Environment: Although High-Performance Computing (HPC) resources were utilized for training, the environment's restrictive administrative policies limited the flexibility of the experimental setup. The research was constrained by the inability to install custom system dependencies, specific libraries, or deploy a fully customizable software stack, thereby preventing the investigation of advanced inference optimizations and real-time deployment simulations.

e. Temporal Constraints: Due to the limited research timeframe of more or less about four months for the researcher to complete all of this research, extensive long-term evaluations could not be conducted within the scope of this work.

CHAPTER V
CONCLUSION

5.1 Conclusion

This research investigates the challenge of fine-grained retail product detection, particularly in scenarios where product variants share nearly identical visual appearances but differ in grammage or size. Based on the problem statements formulated in Chapter 1, the conclusions of this study are presented in the following order.

First, addressing the difficulty of distinguishing fine-grained product variants with highly similar packaging designs, the experimental results show that baseline object detection models frequently suffer from intra-family misclassification. Through optimized training and ensemble strategies, the proposed approach reduces confusion between adjacent grammage variants, indicating that the models are better able to capture subtle visual cues that are critical for fine-grained classification.

Second, concerning the limitations caused by data scarcity and class imbalance, the application of Context-Aware Copy-Paste augmentation improves model generalization on minority classes. Confusion matrix analysis demonstrates a reduction in missed detections for underrepresented variants, although minor inter-class confusion remains. This outcome suggests that data-centric augmentation plays an important role in mitigating the impact of limited training data in real-world retail environments.

Third, with respect to the absence of a systematic optimization strategy, hierarchical hyperparameter tuning and structured post-processing significantly enhance model convergence and prediction consistency. For YOLOv12, the global average False Negative Rate (FNR) decreases from 20.71% in the baseline configuration to 9.70% after optimization. Similarly, Faster R-CNN shows a reduction in FNR from 37.40% to 12.62%. These improvements are accompanied by higher detection accuracy, with optimized YOLOv12 achieving a mAP50:95 of 0.861 and Faster R-CNN reaching 0.859, demonstrating more stable convergence and improved localization performance. Furthermore, the ensemble of both models using Weighted Boxes Fusion (WBF) achieves the lowest global FNR of 7.58% and the highest overall detection performance with a mAP50:95 of 0.905.

The results indicate that the proposed optimization-driven detection pipeline effectively improves the ability of automated retail systems to distinguish visually similar product variants

under data-constrained conditions. By integrating architectural diversity, systematic optimization, data-centric augmentation, and ensemble post-processing, this research demonstrates a practical and scalable solution for fine-grained retail product detection. The overall findings confirm that the proposed approach provides meaningful improvements over baseline detection configurations.

5.2 Recommendations for Future Work

Based on the limitations identified in the preceding section and the potential avenues for expanding the framework's capabilities, the following recommendations are proposed for future research and development:

a. Refining Data Protocols and Collection for Generalization: Future work should focus on substantially increasing the total of data. A larger dataset is essential not only to further boost detection performance but also to enable the adoption of a more robust three-subset split protocol (Train-Validation-Test), thereby reducing potential optimization bias and validating generalization on truly unseen data. K-fold cross validation is also worth to try in the training stage if data is still being considered limited, as it is one of the methods to prevent excessive model bias. Furthermore, data collection efforts may prioritize greater environmental diversity, capturing images across varied lighting conditions, camera angles, and degrees of product occlusion to better reflect real-world shelf chaos.

b. Broadening Domain Scope and Packaging Form Factors: To maximize the practical applicability of the framework, future studies should extend the dataset beyond powdered milk products. Research should incorporate diverse product families, such as bottled drinks, canned goods, snacks, or items with irregular packaging. This expansion requires adapting the feature extraction and augmentation strategies to handle the unique challenges presented by different material reflections and aspect ratios.

c. Exploring Advanced Optimization Methods: Based on the environmental limitations encountered in this study, future researchers are encouraged to explore more advanced optimization tools and methods:

    1. Hyperparameter Exploration and Ablation Studies: Conduct systematic ablation experiments to better understand the influence of individual hyperparameters prior to automated optimization. In this study, hyperparameter tuning is performed using the Optuna framework with a search space closely centered around baseline configurations. While this approach ensures stability, broader ablation-based

exploration may provide stronger insights into parameter sensitivity and facilitate the definition of more informed search spaces before applying automated optimization techniques such as Bayesian-based methods.

2. Hyperparameter Search: Investigate alternative hyperparameter optimization algorithms, such as Bayesian Optimization Hyperband (BOHB), that is claimed to be the state-of-the-art in this context, which were constrained in the researcher's environment but offer potential for faster and more comprehensive convergence.
3. Dataset-Level Strategy: Future research could explore a class-balancing augmentation strategy, by utilizing Copy-Paste to equalize instance counts across all categories. Furthermore, the current Context-Aware Copy-Paste (CACP) method could be integrated with Multi-Scale Copy-Paste method mentioned in Chapter 2. Finally, beyond Copy-Paste strategies, future investigations should explore entirely different dataset-level paradigms to further diversify training scenarios and unlock new improvements in generalization capabilities.
4. Arcitectural Diversity: Utilize comprehensive object detection frameworks such as MMDetection or Detectron2, as these platforms enable the exploration of deeper Faster R-CNN backbones like ResNet-101, ResNeXt-101, and so on. Furthermore, adopting advanced multi-stage detectors like Cascade R-CNN is recommended. Also, investigating entirely different paradigms, particularly Transformer-based models like DETR (Detection Transformer), could potentially offer superior global feature aggregation mechanisms compared to traditional CNN-based approaches.

d. Consistent Evaluation Strategy Across Training and Final Assessment: Future work may consider applying a more consistent evaluation strategy across all training and assessment stages. In this study, baseline modeling and hyperparameter optimization primarily rely on validation-based evaluation, while the final performance assessment is conducted using inference-based evaluation with standardized parameters. Although validation and inference results generally resemble each other, differences in evaluation methodology may lead to apparent performance discrepancies across experimental stages. Aligning evaluation methods, or explicitly incorporating inference-based evaluation throughout the training process, may help reduce potential confusion and improve the clarity of performance interpretation for readers.

e. Developing a Multi-Stage Classification Framework: Future work should explore the implementation of a hierarchical multi-stage pipeline. Unlike the current approach which

relies on a unified detection pipeline, a multi-stage strategy could utilize the optimized detector model primarily for localization, followed by a dedicated, classification model to refine the identification of specific product variants. This separation of tasks offers a promising avenue for further resolving the ambiguities between visually identical SKUs that a single-stage model might miss.

REFERENCES

Aghabagherloo, A., Abadi, A., Sarkar, S., Dasu, V. A., & Preneel, B. (2025, April 17). *Impact of Data Duplication on Deep Neural Network-Based Image Classifiers: Robust vs. Standard Models*. arXiv. https://doi.org/10.48550/arXiv.2504.00638

Ailet. (2025). Retail Scalability Issues in 2025: Automating to Meet the Challenges. Retrieved November 30, 2025, from https://ailet.com/articles/problemas-de-escalabilidad-del-retail-en-2025-automatizar-para-afrontar-los-desafios/

Akiba, T., Sano, S., Yanase, T., Ohta, T., & Koyama, M. (2019, July 25). *Optuna: A Next-generation Hyperparameter Optimization Framework*. arXiv. https://doi.org/10.48550/arXiv.1907.10902

Bartz-Beielstein, T., & Zaefferer, M. (2023). Hyperparameter Tuning Approaches. In E. Bartz, T. Bartz-Beielstein, M. Zaefferer, & O. Mersmann (Eds.), *Hyperparameter Tuning for Machine and Deep Learning with R: A Practical Guide* (pp. 71–119). Singapore: Springer Nature. https://doi.org/10.1007/978-981-19-5170-1_4

Barz, B., & Denzler, J. (2020). Do We Train on Test Data? Purging CIFAR of Near-Duplicates. *Journal of Imaging*, *6*(6), 41. https://doi.org/10.3390/jimaging6060041

Bergstra, J., Bardenet, R., Bengio, Y., & Kégl, B. (2011). Algorithms for Hyper-Parameter Optimization. *Advances in Neural Information Processing Systems*, *24*. Curran Associates, Inc. Retrieved from https://papers.nips.cc/paper_files/paper/2011/hash/86e8f7ab32cfd12577bc2619bc635690-Abstract.html

Bergstra, J., & Bengio, Y. (2012). Random Search for Hyper-Parameter Optimization. *Journal of Machine Learning Research*, *13*(10), 281–305.

Bizplanr Team. (2025, May 30). Top 30+ Retail Industry Statistics You Must Know in 2025. Retrieved December 13, 2025, from Bizplanr website: https://bizplanr.ai/blog/retail-industry-statistics

Bochkovskiy, A., Wang, C.-Y., & Liao, H.-Y. M. (2020, April 23). *YOLOv4: Optimal Speed and Accuracy of Object Detection*. arXiv. https://doi.org/10.48550/arXiv.2004.10934

Bodla, N., Singh, B., Chellappa, R., & Davis, L. S. (2017, August 8). *Soft-NMS -- Improving Object Detection With One Line of Code*. arXiv. https://doi.org/10.48550/arXiv.1704.04503

Budimir, L. A., Kalafatić, Z., Subašić, M., & Lončarić, S. (2025). Context-Aware Fine-Grained Product Recognition on Grocery Shelves. *IEEE Access*, *13*, 16824–16837. https://doi.org/10.1109/ACCESS.2025.3532745

Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., & Zagoruyko, S. (2020, May 28). *End-to-End Object Detection with Transformers*. arXiv. https://doi.org/10.48550/arXiv.2005.12872

CLRN team. (2025, July 2). How to handle duplicate data in machine learning? Retrieved January 7, 2026, from California Learning Resource Network website: https://www.clrn.org/how-to-handle-duplicate-data-in-machine-learning/

Dipto, I. C., Cassidy, B., Kendrick, C., Reeves, N. D., Pappachan, J. M., Chandrabalan, V., & Yap, M. H. (2023). *Quantifying the Effect of Image Similarity on Diabetic Foot Ulcer Classification*. https://doi.org/10.1007/978-3-031-26354-5_1

Dollar, P., Wojek, C., Schiele, B., & Perona, P. (2012). Pedestrian Detection: An Evaluation of the State of the Art. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *34*(4), 743–761. https://doi.org/10.1109/TPAMI.2011.155

Falkner, S., Klein, A., & Hutter, F. (2018). BOHB: Robust and Efficient Hyperparameter Optimization at Scale. *Proceedings of the 35th International Conference on Machine Learning*, 1437–1446. PMLR. Retrieved from https://proceedings.mlr.press/v80/falkner18a.html

Franceschi, L., Donini, M., Perrone, V., Klein, A., Archambeau, C., Seeger, M., … Frasconi, P. (2025). Hyperparameter Optimization in Machine Learning. *Foundations and Trends® in Machine Learning*, *18*(6), 1054–1201. https://doi.org/10.1561/2200000088

Ganaie, M. A., Hu, M., Malik, A. K., Tanveer, M., & Suganthan, P. N. (2022). Ensemble deep learning: A review. *Engineering Applications of Artificial Intelligence*, *115*, 105151. https://doi.org/10.1016/j.engappai.2022.105151

Ghiasi, G., Cui, Y., Srinivas, A., Qian, R., Lin, T.-Y., Cubuk, E. D., … Zoph, B. (2021, June 23). *Simple Copy-Paste is a Strong Data Augmentation Method for Instance Segmentation*. arXiv. https://doi.org/10.48550/arXiv.2012.07177

Girshick, R. (2015, September 27). *Fast R-CNN*. arXiv. https://doi.org/10.48550/arXiv.1504.08083

Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014, October 22). *Rich feature hierarchies for accurate object detection and semantic segmentation*. arXiv. https://doi.org/10.48550/arXiv.1311.2524

Grand View Research. (2025, June). Computer Vision AI in Retail Market | Industry Report, 2033. Retrieved December 13, 2025, from https://www.grandviewresearch.com/industry-analysis/computer-vision-ai-retail-market-report

Guo, P., Yang, H., & Sano, A. (2023, September 18). *Empirical Study of Mix-based Data Augmentation Methods in Physiological Time Series Data*. arXiv. https://doi.org/10.48550/arXiv.2309.09970

Guo, Q. (2025, May 21). *Enrich the content of the image Using Context-Aware Copy Paste* (Version 2). Version 2. arXiv. https://doi.org/10.48550/arXiv.2407.08151

Harness Team. (2025, December 17). What is False Positive Rate? Retrieved December 24, 2025, from Harness.io website: https://www.harness.io/harness-devops-academy/false-positive-rate

Huang, J., Rathod, V., Sun, C., Zhu, M., Korattikara, A., Fathi, A., … Murphy, K. (2017, April 25). *Speed/accuracy trade-offs for modern convolutional object detectors*. arXiv. https://doi.org/10.48550/arXiv.1611.10012

Hussein, B. M., & Shareef, S. M. (2024). An Empirical Study on the Correlation between Early Stopping Patience and Epochs in Deep Learning. *ITM Web of Conferences*, *64*, 01003. https://doi.org/10.1051/itmconf/20246401003

Jiang, B., Wang, J., Ren, G., Zhi, M., Yu, Z., Zhang, Y., … Jia, S. (2025). Research on pedestrian detection method based on multispectral intermediate fusion using YOLOv7. *Scientific Reports*, *15*(1), 16851. https://doi.org/10.1038/s41598-025-88871-y

Jocher, G., Chaurasia, A., & Qiu, J. (2023). Ultralytics YOLO. Retrieved January 7, 2026, from GitHub website: https://github.com/ultralytics/ultralytics/blob/main/ultralytics/cfg/default.yaml

John, A. C. (2025, January 14). Mastering Duplicate Data Management in Machine Learning for Optimal Model Performance. Retrieved January 7, 2026, from DagsHub Blog website: https://dagshub.com/blog/mastering-duplicate-data-management-in-machine-learning-for-optimal-model-performance/

Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A., & Talwalkar, A. (2018, June 18). *Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization*. arXiv. https://doi.org/10.48550/arXiv.1603.06560

Lin, T.-Y., Goyal, P., Girshick, R., He, K., & Dollár, P. (2018, February 7). *Focal Loss for Dense Object Detection*. arXiv. https://doi.org/10.48550/arXiv.1708.02002

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., & Berg, A. C. (2016). *SSD: Single Shot MultiBox Detector*. https://doi.org/10.1007/978-3-319-46448-0_2

Market.us. (2024, February). AI in Retail Market. Retrieved December 14, 2025, from Market.us website: https://market.us/report/ai-in-retail-market/

Miller, D., Moghadam, P., Cox, M., Wildie, M., & Jurdak, R. (2022). What's in the Black Box? The False Negative Mechanisms Inside Object Detectors. *IEEE Robotics and Automation Letters*, *7*(3), 8510–8517. https://doi.org/10.1109/LRA.2022.3187831

Oksuz, K., Cam, B. C., Akbas, E., & Kalkan, S. (2018, July 5). *Localization Recall Precision (LRP): A New Performance Metric for Object Detection*. arXiv. https://doi.org/10.48550/arXiv.1807.01696

Peng, J., Xiao, C., & Li, Y. (2021, September 1). *RP2K: A Large-Scale Retail Product Dataset for Fine-Grained Image Classification*. arXiv. https://doi.org/10.48550/arXiv.2006.12634

Pettersson, T., Riveiro, M., & Löfström, T. (2024). Multimodal fine-grained grocery product recognition using image and OCR text. *Machine Vision and Applications*, *35*(4), 79. https://doi.org/10.1007/s00138-024-01549-9

Ramos, P., Ramos, R., & Garcia, N. (2025, August 24). *Data Leakage in Visual Datasets* (Version 1). Version 1. arXiv. https://doi.org/10.48550/arXiv.2508.17416

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016, May 9). *You Only Look Once: Unified, Real-Time Object Detection*. arXiv. https://doi.org/10.48550/arXiv.1506.02640

Ren, S., He, K., Girshick, R., & Sun, J. (2016, January 6). *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*. arXiv. https://doi.org/10.48550/arXiv.1506.01497

Rumala, D. J. (2023). *How You Split Matters: Data Leakage and Subject Characteristics Studies in Longitudinal Brain MRI Analysis*. https://doi.org/10.1007/978-3-031-45249-9_23

Saputra, H., Muchtar, K., Chitraningrum, N., Andria, A., & Febriana, A. (2025). Performance evaluation of hyper-parameter tuning automation in YOLOV8 and YOLO-NAS for corn leaf disease detection. *SINERGI*, *29*(1), 197–206. https://doi.org/10.22441/sinergi.2025.1.018

Shah, D. (2022, March 7). Mean Average Precision (mAP) Explained: Everything You Need to Know. Retrieved December 2, 2025, from https://www.v7labs.com/blog/mean-average-precision

Snoek, J., Larochelle, H., & Adams, R. P. (2012). Practical Bayesian Optimization of Machine Learning Algorithms. *Advances in Neural Information Processing Systems*, *25*. Curran Associates, Inc. Retrieved from https://papers.nips.cc/paper_files/paper/2012/hash/05311655a15b75fab86956663e1819cd-Abstract.html

Solovyev, R., Wang, W., & Gabruseva, T. (2021). Weighted boxes fusion: Ensembling boxes from different object detection models. *Image and Vision Computing*, *107*, 104117. https://doi.org/10.1016/j.imavis.2021.104117

Srivastava, M. M., & Kumar, P. (2021, October 7). *Machine Learning approaches to do size based reasoning on Retail Shelf objects to classify product variants*. arXiv. https://doi.org/10.48550/arXiv.2110.03783

Sumanth, P. (2025, December 5). Planogram Compliance: How to Run Visual Merchandising Audits with Photo Proof. Retrieved December 13, 2025, from https://taqtics.co/retail-operations/planogram-compliance/

Tian, Y., Ye, Q., & Doermann, D. (2025, February 18). *YOLOv12: Attention-Centric Real-Time Object Detectors*. arXiv. https://doi.org/10.48550/arXiv.2502.12524

Wu, Y., Kirillov, A., Massa, F., Lo, W.-Y., & Girshick, R. (2019). Detectron2. Retrieved January 7, 2026, from GitHub website: https://github.com/facebookresearch/detectron2/blob/main/detectron2/config/defaults.py

Yu, T., & Zhu, H. (2020, March 12). *Hyper-Parameter Optimization: A Review of Algorithms and Applications*. arXiv. https://doi.org/10.48550/arXiv.2003.05689

Yu, X., Li, F., Liu, Y., & Wang, A. (2023). A Multi-Stage Adaptive Copy-Paste Data Augmentation Algorithm Based on Model Training Preferences. *Electronics*, *12*(17), 3695. https://doi.org/10.3390/electronics12173695

Yun, S., Han, D., Oh, S. J., Chun, S., Choe, J., & Yoo, Y. (2019, August 7). *CutMix: Regularization Strategy to Train Strong Classifiers with Localizable Features*. arXiv. https://doi.org/10.48550/arXiv.1905.04899

Zhang, H., Cisse, M., Dauphin, Y. N., & Lopez-Paz, D. (2018, April 27). *mixup: Beyond Empirical Risk Minimization*. arXiv. https://doi.org/10.48550/arXiv.1710.09412

Zhao, Y., Lv, W., Xu, S., Wei, J., Wang, G., Dang, Q., … Chen, J. (2024, April 3). *DETRs Beat YOLOs on Real-time Object Detection*. arXiv. https://doi.org/10.48550/arXiv.2304.08069

Zhao, Z.-Q., Zheng, P., Xu, S., & Wu, X. (2019, April 16). *Object Detection with Deep Learning: A Review*. arXiv. https://doi.org/10.48550/arXiv.1807.05511

Zhu, X., Su, W., Lu, L., Li, B., Wang, X., & Dai, J. (2021, March 18). *Deformable DETR: Deformable Transformers for End-to-End Object Detection*. arXiv. https://doi.org/10.48550/arXiv.2010.04159

APPENDIX