AUTOMATED IMAGING SYSTEM FOR MICROPLASTICS SAMPLE CLASSIFICATION USING CONVOLUTIONAL NEURAL NETWORK

An Undergraduate Thesis presented to the
School of Engineering
of the University of San Carlos

In partial fulfillment
of the requirements for the degree
Bachelor of Science in Computer Engineering

Piolo Pascual E. Besinga
Ivor Louisetyne Canque
Lysander S. Uy

Main Adviser: Dr. Luis Gerardo S. Cañete, Jr.

Assistant Adviser: Dr. Philip Virgil B. Astillo

December 2025

Declaration

The authors hereby declare that, except where explicit reference is made to the work of others, the content of this thesis is original and has not been submitted, in whole or in part, for consideration towards any other degree or qualification at this or any other university. To the best of our knowledge and belief, this thesis contains no material authored by another person, whether published or unpublished, nor does it include any falsified or fabricated content. This thesis represents our independent work, with no contributions from external collaborators unless specifically acknowledged in the content and acknowledgments sections.

Piolo Pascual E. Besinga
Ivor Louisetyne Canque
Lysander S. Uy
December 2025

Acknowledgments

First and foremost, we offer our deepest praise and gratitude to the Almighty God for His wisdom, and guidance throughout this entire research journey. His constant presence provided us with the strength to overcome challenges, the resilience to rise above failures, and the clarity of mind to persevere during moments of difficulty, making the completion of this research possible.

We extend our sincere appreciation to our adviser, Dr. Luis Gerardo S. Cañete, and assistant adviser, Dr. Philip Virgil B. Astillo, for their mentorship and guidance. Their technical insights, patience, and constructive feedback were instrumental in shaping the direction of this study. We are deeply grateful for their willingness to share their expertise and for pushing us to achieve a higher standard of work.

To the distinguished members of the panel, Dr. Rosana J. Ferolin, Engr. Elline L. Fabian and Engr. James Michael C. Cañete, thank you for your time, thorough review, and insightful critiques. Your constructive suggestions and probing questions significantly improved the quality of this research. Your expertise helped refine the methodology and ensured the accuracy of the results presented in this paper.

A special thanks goes to our seniors, for their mentorship and willingness to lend a helping hand. Their practical advice, technical tips, and shared experiences provided a roadmap that made the complexities of this project more manageable. Their encouragement served as a constant reminder that success is achievable.

Finally, we express our profound gratitude to our friends, colleagues, and families for their unwavering love and emotional support. To our colleagues, thank you for the camaraderie and collaboration that made the work lighter. To our friends and families, thank you for your understanding, prayers, and encouragement during the long nights and stressful periods. You were our inspiration to reach the finish line.

Abstract

The University of San Carlos developed a microplastic imaging system, C.Scope.AI, which has proven to address the challenges of microplastic microscopy. Despite its efficacy, the system presented gaps, as its software is fragmented, the XY platform is overspecified, and the system has a complicated retraining. To address these issues, Version 2 introduced a redesigned hardware and software architecture. Key improvements include a hardware design that is adaptable to more existing microscopes, a simplified software architecture for easier setup to end users with non-technical skills, and a retraining pipeline that enables local model retraining to improve generalization. The redesigned system demonstrated significant physical and economic efficiency gains, resulting in a unit that is **42%** smaller than Version 1, with a cost reduction from 14,500.00 PHP to 8,934.00 PHP. Furthermore, software performance from benchmarks on an Intel(R) Core(TM) i3-1115G4 system revealed detection speedups of 9.8% and 11.71% for binary and multiclass models, respectively, alongside a reduction in memory utilization of **~1,459.4 MB (1.5 GB)**. Lastly, The local retraining pipeline demonstrated an improved generalization on unseen data by 3.6% (relative gain of AP), specifically improving the detection of small-scale microplastics with confidence levels exceeding 90%.

Table of Contents

List of Figures

List of Tables

Nomenclature

Acronyms / Abbreviations

ANN Artificial Neural Network

AP Average Precision

API Application Programming Interface

ASEAN Association of Southeast Asian Nations

CAD Computer-Aided Design

CNN Convolutional Neural Network

COCO Common Objects in Context (Dataset Format)

CPU Central Processing Unit

DNN Deep Neural Network

FN False Negative

FP False Positive

FPN Feature Pyramid Network

FTIR Fourier Transform Infrared Spectroscopy

GPU Graphics Processing Unit

GRBL G-code Sender and Motion Controller Firmware

HERR Hard Example Reduction Rate

HPO Hyperparameter Optimization

IoU Intersection over Union

LLM Large Language Model

LR Learning Rate

mAP Mean Average Precision

MLR Mean Loss Reduction

MP Microplastic

MSDs Musculoskeletal Disorders

NMS Non-Maximum Suppression

ROI Region of Interest

RPN Region Proposal Network

SEA-MaP Southeast Asia Program on Microplastic

SLM Small Language Model

TPE Tree-structured Parzen Estimator

TP True Positive

UI User Interface

V1 Version 1

V2 Version 2

WSL Windows Subsystem for Linux

XY Horizontal Cartesian Coordinates

Chapter 1

Introduction

Plastic, one of the most important inventions of the last century, transformed the world in ways never seen before. The advent of plastic enabled the facets of modern life, from food packaging to electronic devices. However, the environmental ramifications have surfaced in an unfavorable way. Plastics are persistent and pervasive throughout its lifespan and have now been reported to be seen from ocean trenches to mountain summits[1]. It is found that Plastic production on a global scale reached nearly 368 million tons in 2019 and is expected to reach 1.1 billion tons by 2050. In tourist destinations, where natural resources play a role in ecological sustainability and economic viability, microplastics are a growing concern [2]. Plastic pollutions are diverse in size, Macroplastics are over 0.5 centimeters (cm), Mesoplastics range from 25 milimeters to 5 milimeters (mm), and Microplastics measuring at less than 5 mm [3]. Microplastics pollute the environment through runoff, accidental releases, sewage discharges, and emissions from residential and industrial waste process [4]. We humans may suffer from oxidative stress, cytotoxicity, neurotoxicity, immune system disruption, and the binding of microplastics to different tissues [5]. Addressing the challenges created by microplastics requires comprehensive strategies, including reducing plastic production, improving waste management, and advancing research on their environmental and health impacts [6].

One of the organizations combating microplastic pollution are the Southeast Asia Program on Microplastic (SEA-MaP) researchers from the SEA-MaP program of the Association of Southeast Asian Nations (ASEAN) [7]. Sea-MaP is a five-year program that aims to contribute to the prevention of land- and sea-based marine microplastic pollution. Endorsed by ASEAN member states, SEA-MaP addresses the pollution issue through strengthening regional policies and institutions for plastic circularity.

In the Philippines, government and academic groups study the presence of microplastics in their own areas. One example would be the Biology Department of the University of

San Carlos, wherein, under the SEA-MaP, they study different samples all around Cebu Province. The Computer Engineering Department has previously collaborated with the Biology Department after identifying a solution to a tedious process with their research that is solvable through computer systems amplified by machine learning. This led to the development of an automated imaging system entitled C.Scope.AI.

Fig. 1. Overall System of C.Scope.AI

Figure 1 illustrates the complete C.Scope.AI system. The system's deployment has significantly reduced the tedious and strenuous work of the SEA-MaP researchers. Having the development of the automated XY Platform addressing the need of manually moving the petridish and the development User Interface (UI) with integrated artificial intelligence streamlines the process of repetitive movement between microscope and camera for microplastic identification.

1.1 Statement of the Problem

The initial iteration demonstrated promising outcomes in the field of microplastic analysis. The machine learning model achieved an accuracy rate of 91% in classifying microplastic samples and received positive feedback from researchers regarding its potential to enhance workflow efficiency. Despite these accomplishments, several critical limitations in the original system were identified that made room for generalization.

One of which is that their hardware module—specifically the XY Platform, is very costly, large in profile, and overkill; doing so much of what it is intended to do. Its over-specialized design resulted in it not being able to adapt and work with most existing microscope setups, posing a significant barrier to widespread use. Another concern is how the software architecture is highly fragmented. The software application relied on numerous external

dependencies, one of which is using Docker to containerize their model, making the installation and configuration process cumbersome which significantly reduced the out-of-the-box experience for end users and required the direct involvement of the development team for a streamlined setup process. Furthermore, the system exhibited a heavy runtime load and contained multiple points of failure, reducing overall reliability. Lastly, their machine learning model lacked the ability to incorporate new datasets dynamically. This issue affects the accuracy skewness of the model, which can hinder its continuous learning and improvement.

1.2 Goal and Objectives

This study's primary goal is to expand usage among Microplastic Researchers doing Microscopy and ease of use for end-users. The previous implementation of C.Scope.AI v1, though proven to be a viable solution that addressed challenges in traditional microplastic microscopy, presents an area for development.

The number of targets identified to ensure an automated process in achieving the main goal are the following:

- Redesign the software architecture to enhance simplicity without relying on external dependencies
- Redesign the automated XY platform with the extensive ability to adapt and cater to existing microscopes.
- Enhance cost-effectiveness while maintaining overall functionality
- Design a Retrainable CNN Architecture for microplastic identification through a Retraining Pipeline.

1.3 Significance of the Study

This study aims to improve the adaptability and usability of C.Scope.AI not only for SEA-MaP researchers only but also facilitating its adoption across a wider range of user applications. One of the significant aspects of the study is to improve the overall user experience by refining the software architecture and streamlining the setup process—potentially making it simpler to a single-click operation. Furthermore, the need for developer intervention during model training has been eliminated through the introduction of user-side retraining capabilities, thereby minimizing workflow complexity and empowering end-users to independently update the system as needed.

1.4 Scope and Limitations

The scope and limitations of this study are outlined as follows:

- This study is specifically centered on enhancing the initial version of the system by redesigning its hardware, software architecture, and the machine learning model
- The system maintains the accuracy performance of the first iteration, with no significant effort directed at further enhancing it.
- Deployment is specifically tailored for microplastic researchers in the University of San Carlos, which limit the system's accessibility to other users that do not deal with microscopy; however, it remains open for potential widespread use in the future
- Adaptability of the system is restricted by the physical dimensions that the XY platform can accommodate, particularly within the limited space available between the microscope's lens and its base.

Chapter 2

Background of Related Works

Microplastics

Amid contemporary society, scientists define microplastic (MP) as plastic particles or debris that is smaller than five (5) millimeters in diameter but larger than one (1) millimeter [8]. Microscopes are the most widely used identification tool for counting, sorting Microplastic particles according to color, size, brightness, and shape [9]. Microplastics are classified into six (6) categories based on its shape: sheet, film, line/fiber, fragment, pellet/granule, and foam [10]. MPs are found in a wide range of locations including aquatic and terrestrial ecosystems due to human activities. [11]. The Philippines ranks as the third largest producer of plastic waste, thereby compounding the issue of Microplastics [12]. MPs are identified as a global threat to human health and the environment. The gradual degradation of plastic wastes into microplastics persists indefinitely in the environment [13].

Microplastics Research

This issue led to researchers using a variety of methods like visual identification to assess how microplastics are present in diverging environments. However, despite exponential research regarding MPs pollution in the preceeding decades, authenticated and standardized methods for sampling, quantification, characterisation lag behind resulting in a hampered interlaboratory comparability [14]. Extraction, separation, identification, and quantification are the steps in the analytical process applied to microplastics in environmental samples [15]. The simplest readmissable analysis protocol involves examination with the naked eye, resulting in error margins as high as 70% when re-analyzed with spectroscopic techniques [16]. A proposed automated technique that utilized mobile phones Digital Signal Lens Reflex (DSLR) was made, but fell through since the use of transformation tools had to

be observed to obtain a perpendicular position of the particles with the camera [17]. The state-of-the-art pre-existing chemical-based technique of pyrolysis or thermal decomposition gas chromatography coupled with mass spectrometry, Fourier transform infrared (FTIR) spectroscopy, or Raman spectroscopy is most prevalent method of microplastic identification, however access to pricey analytical equipment deem it not feasible [18]. This complex issue which is the top mitigation precedence combined with an empty space for a unified classification process encourages researchers to create technological advancements that can easily detect and classify microplastics.

Microplastics research is an ever-evolving field, with scientists worldwide contributing their unique expertise to address the growing and pervasive presence of microplastics across our planet. This area of research lacks a standardized approach that accommodates researchers with varying situational constraints, such as financial and technological limitations.

Occupational Concerns with Prolonged Use of Microscopes and its Effects

Microscope work can be physically demanding, exerting strain on both visual and musculoskeletal systems. Using a microscope for a prolonged duration imposes a great concern for health and ergonomics [19]. Microscope users may experience Musculoskeletal Disorders (MSDs) for several reasons: exerting much force, repetitive movements, and awkward and prolonged static postures [20]. Through surface electromyography measurement, it was observed that the neck muscles generate more electrical activity during cognitive tasks using a microscope [21]. They also experience visual discomfort due to repetitive eye movements, aggravation of ametropia and may experience stress and burnout due to intensive work hours[22]. A study from [23] found that prolonged use of microscopes can cause fatigue in the neck and back after 5.8 min and 8.4 min of usage, respectively. The same study also reported that consecutive utilization of microscopes led to neck pain after 14.1 min and back pain after 13.7 min on average. Another study shows that 62% of the participants mentioned that they suffered from musculoskeletal problems and 56% took regular short breaks. It was determined that extended use of microscopes is significantly associated with and contributes to musculoskeletal issues. [22]

Moreover, workers engage in intensive visual labor, as continuous eye movement is required to clearly observe samples when using a traditional microscope. Prolonged use of a microscope can contribute to visual fatigue. A growing concern is that visual fatigue may lead to a decreased attention span and reduced vigilance in the user [24]. According

to studies by [25][26] about 80% of full-time microscope operators experienced various symptoms of visual strain. The most common symptoms that the users met were burning of the eye, teary eyes, headache, and eye pain. Another study by [27] saw that by applying visual criteria for screening in categories of visual acuity, eye muscle balance and depth perception saw 14% of microscope operators were transferred to other jobs due to failing to meet the criteria. In the study of [28], These findings indicate a positive effect and direct correlation between visual fatigue and working hours, suggesting that prolonged work hours contribute to increased visual fatigue.

In conclusion, prolonged microscope use poses significant health concerns, particularly musculoskeletal disorders and visual fatigue. In addition, it can also cause stress and anxiety because of excessive labor hours. Having these risks mentioned, it can affect researchers' performance and productivity, and also the willingness to do microscopy.

Convolutional Neural Networks Capabilities

Artificial Neural Networks (ANNs) are computational processing systems of which are heavily inspired by way biological nervous systems (such as the human brain) operate. Convolutional Neural Networks (CNNs) are analogous to traditional ANNs in that they are comprised of neurons that self-optimise through learning. Each neuron will still receive an input and perform a operation (such as a scalar product followed by a non-linear function) - the basis of countless ANNs [29].CNN learns from data using backpropagation and optimization algorithms, making it similar to other neural networks. However, its main distinction lies in the fact that a Convolutional Neural Network is built for spatial data and its ability to preserve spatial structures. This type of neural network algorithm leverages spatial locality and translational invariance by making use of a learnable filter that slides across a pixel represented image regionally rather than analyzing the entire image at once. This smaller matrix composed of weights and bias is applied equally among each region ensuring that the same features are captured regardless of their position.

CNN is a type of deep learning model for processing data that has a grid pattern, such as images, which is inspired by the organization of animal visual cortex and designed to automatically and adaptively learn spatial hierarchies of features, from low- to high-level patterns [30] [31]. This overall structure of a CNN ensures accurate predictions especially when dealing with image processing. This is why Convolutional Neural Network is the most established algorithm among deep learning models, especially in computer vision [32].

Convolutional Neural Networks in Microplastic Detection and Classification

Convolutional Neural Network offers a strong accuracy in the area of literature concerned about microplastics identification [33]. Currently, the stereo microscope is the most frequently used tool in identifying microplastics according to their shape, form, brightness, color [34]. However, challenges remain since these studies were unable to create a uniform classification of microplastic data for the various microplastics in nature. In microplastics analysis, physical features are a crucial aspect. However, determining key properties such as size, and shape require time-consuming and intensive manual labour which would raise the need for an automated image analysis that helps with classification using deep learning neural networks and computer image processing. Given that CNNs can be trained to extract features from images, microplastics samples in this case, this architecture of deep learning neural network offers a viable solution that would allow for faster and more accurate analysis of microplastic sediments [15].

Early studies demonstrated the capability of CNN in microplastic identification. [35] achieved 89% accuracy in wastewater microbead classification, while Lihui, Shuang, and Shi et. al utilized an enhanced approach integrating CNN with Micro-Raman Spectroscopy, achieving even higher precision [36]. CNN has shown excellent performance in many computer vision and machine learning applications. Many credible papers have been published on the topic and multiple high quality open source CNN software packages are available [37].

While CNN alone has proven effective, comparative studies further highlight its superiority over traditional machine learning models. A study by [38] conducted comparison between Artificial Neural Network, Random Forest, Deep Neural Network, and Convolutional Neural Network, a variety of machine learning models by testing on aged microplastics and commercial microplastics to identify which type of polymer the samples are. The identification results for commercial microplastics achieved 50% for ANN, 92.9% for RF, 100% for DNN, and 100% for CNN, respectively. For aged microplastics, results achieved 40% for ANN, 60% for RF, 80% for DNN, and 100% for CNN. Similarly, a study by [39] used

Author	Paramaters	Input	Method	Accuracy(%)
Bernales et. al	Polymer	RGB	Faster-RCNN +ResNet-50-FPN	92.4 (binary) 86.0 (multi-class)
Thammasanya et. al	Polymer type	RGB	Faster-RCNN +ResNet-50-FPN	87.8
Yurtsever et. al	Shape, size, and texture	RGB	CNN	89.0
Navarro et. al	Shape, size, and texture	RGB	U-Net &VGG16	98.11
Lihui et. al	Chemical Structure	Spectra	CNN	96.43
Zhang et. al	Chemical Structure	Spectra	1D-CNN KNN SVM MLP AlexNet	96.4 93.57 94.16 94.16 93.4
Zeng et. al	Polymer	Spectra	ANN RF DNN CNN	40.0 60.0 80.0 100.0

Table I. Comparison of Different Classification Methods Recognition Accuracies

another variation of CNN, 1D-CNN together with Raman spectroscopy and tested on raw and pre-processed data. The model was able to achieve 96. 4% precision for raw data and 96. 2% for preprocessed data. Further, the same datasets were trained on different models to compare with their CNN model. Results show 92.57% with raw data and 93.57% with pre-processed data for K-Nearest Neighbors , while the Support Vector Machine reached 88.43% and 94.16%, respectively. The Multi-Layer Perceptron recorded 90.32% for raw data and 92.58% for pre-processed data. AlexNet showed 94.02% accuracy with raw data and 93. 4% with preprocessing, while 1D-CNN achieved the highest accuracy at 95.82% for raw data and 95. 47% for preprocessed data.

Beyond conventional CNN models, more advanced architectures like Faster-RCNN with ResNet-50 have been explored to enhance feature extraction. A study by [15] achieved 87.8% by incorporating this approach. [19] further optimized the method using a Feature Pyramid Network, increasing accuracy to 92.4% on binary, and 86% on multiclass scenarios.v

The continuous refinement of CNN-based approaches culminates in a more advanced method: a two phase approach by [40] wherein a U-Net network was implemented first for the image segmentation. Afterwhich, the images were classified using a VGG16 network. The study was able to achieve a high accuracy of 98.11%.

Table I (*see Table I above for reference*) summarizes these related works confirm that machine learning models, such as Convolutional Neural Networks (CNN), can accurately identify and classify microplastics with high precision. Deep learning architectures with CNNs can significantly enhance image classification, particularly when multiple phases and model combinations are utilized to optimize accuracy.

Retraining on Convolutional Machine Learning Models

Data is one of the most essential component of a machine learning model. Contemporary convolutional machine learning models are usually pre-trained during development, unable to intake post-deployment data. These systems are fine-tuned and optimized for optimal performance and is a tedious and mentally exhausting process. This approach heavily focuses on the development with no attention given to performance after deployment essentially making it a machine learned model. To tackle the gap, model retraining is added in the equation. Model retraining contains multiple approaches ranging from real-time learning to scheduled updating.

One way to enable retraining is by automating the machine learning pipeline. This involves having methods for streamlining data annotation, choosing the optimal hyper parameters, and deploying the model. A study by [41] automated their retraining process using an automating tool called AutoML [42], eliminating the need to create a model from the user and automatically choosing the optimal values of the hyperparameter. They did the automating process while offloading the compute usage to the cloud. TPOT, a package in Python was used to assist in automating [43]. The study of [41] addresses the issue of retraining model being complex and time-consuming. Through giving the model the capability to detect change in nature of data and then retrain all by itself, they addressed the issue. They concluded that retraining may take longer than the pre-deployment training process, however it improves the accuracy of results.

Another study by [44] focuses on the model retraining on cyber-physical systems like autonomous vehicles. In their work, the researchers proposed AgileML [45], an end-to-end

retraining framework that monitors runtime data drift and automatically triggers retraining processes to ensure model relevance over time. Their approach integrates lightweight drift detectors and selectively retrains models based on the severity of drift observed. Unlike conventional full retraining, AgileML introduces delta retraining to minimize retraining overhead by reusing parts of the existing model and adapting only to new data. This makes it especially suitable for edge devices with constrained resources. Their experiments on autonomous driving datasets show that AgileML improved model adaptability while maintaining low computational overhead. The study emphasizes the importance of continuous adaptation in mission-critical AI systems and shows that proactive retraining leads to safer and more reliable deployment in dynamic environments.

Chapter 3

Hardware Architecture

This chapter presents the development and implementation of the redesigned XY platform. It outlines the design constraints, key considerations, and essential changes while examining important aspects such as size, cost, ease of replication, and adaptability.

3.1 The Problem

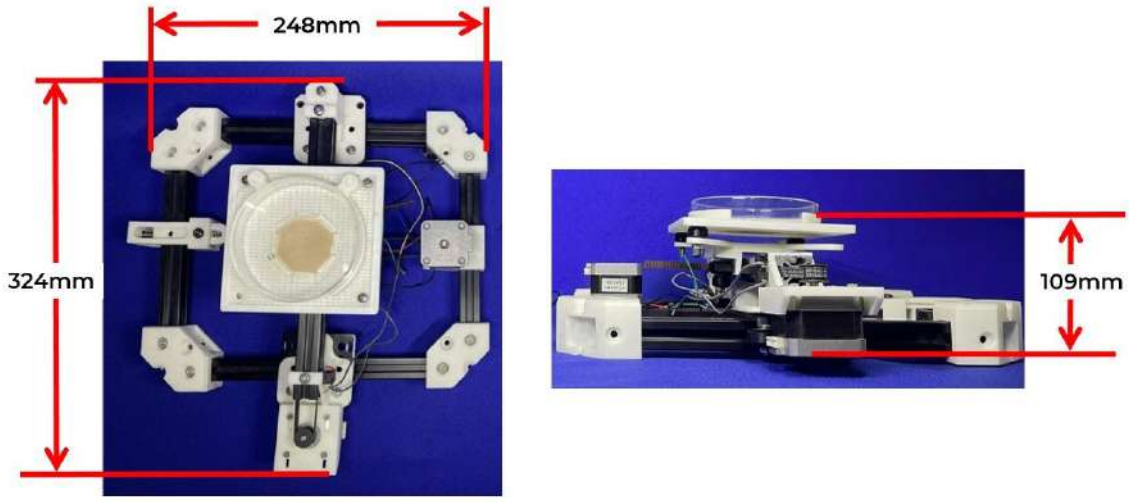


Fig. 2. XY Platform V1 Dimensions

3.1.1 Excessive Physical Footprint

First, the previous platform seemed significantly oversized for its micro-movement application, creating practical difficulties for end users. Its stacked-gantry mechanism relied on multiple layers of thick aluminum extrusions, large brackets, and mounting plates,

resulting in a bulky, heavy assembly that was both vertically tall and horizontally massive. While this architecture allowed independent axis motion, the oversized design introduced several challenges:

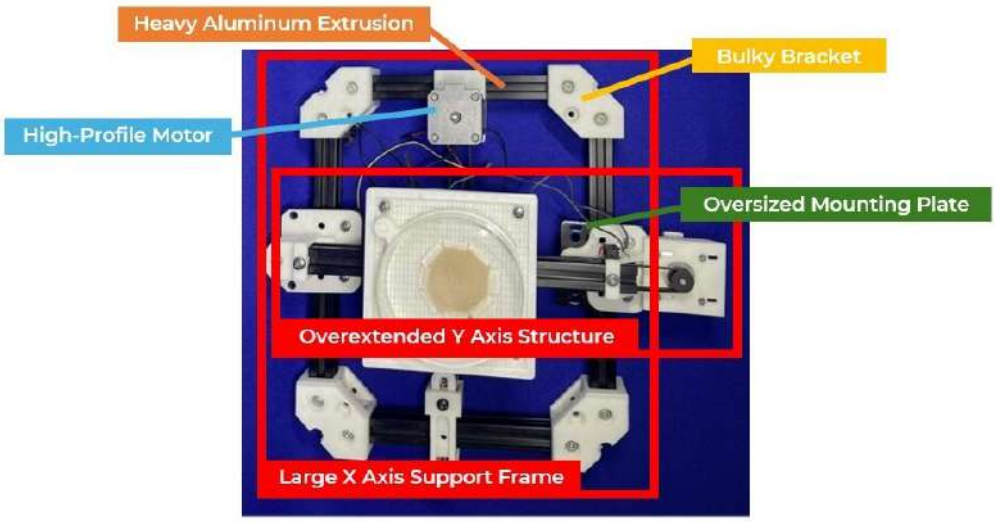


Fig. 3. Identifying Mechanical Gaps of XY Platform V1

Overspecification of materials

Industrial-grade aluminum extrusions and heavy brackets added unnecessary mass and rigidity, far exceeding the mechanical requirements for moving only a Petri dish.

Redundant structural stacking

Multiple support layers for the X and Y axes created a complex assembly with duplicated components, which further increased the platform's size without improving functionality.

Inefficient use of work volume

Despite its large size, only a small central region of the platform was actively used for imaging. Most of the space added no benefit, creating a heavy, unwieldy system that required significant bench space and complicated placement and operation for users.

Limited adaptability

The stacked-gantry arrangement increased the hardware's height profile. Combined with its overall bulk, this design restricted integration with existing digital microscopes because its physical layout was not adaptable enough to fit within some laboratory setups.

3.1.2 High Fabrication Cost

Another limitation of the XY Platform Version 1 was its high fabrication cost, which was closely tied to hardware design choices. The platform relied on heavy aluminum extrusions for the gantry, multiple linear wheel guides, mounting plates, and closed-loop GT2 belts. These components, while functional, were inherently expensive due to their size, materials, and specifications. Additionally, some parts of the entire hardware, beyond the platform itself, were purchased from higher-priced suppliers, which further increased the total cost. This elevated cost limited adoption among laboratories with restricted budgets and reduced the practicality of the system for widespread use.

3.1.3 Poor Replicability

Lastly, the XY Platform Version 1 was difficult to replicate due to its mechanically complex stacked-gantry design, which required precise alignment and placement of certain components. Successfully assembling the system required more than minimal mechanical knowledge, as even minor misalignments could compromise performance. In practice, researchers who wanted a working platform often faced challenges with independent replication. The physical complexity and interdependence of components added difficulty for end users who focus primarily on microscopy rather than engineering. Consequently, replicating Version 1 was labor-intensive and limited accessibility, emphasizing the need for a simpler, more user-friendly design in subsequent versions.

3.2 Solution

After identifying key overlooked areas in the hardware, the Version 2 redesign focused on the following objectives: reducing the physical footprint and lowering overall costs while improving replicability and ease of use for researchers. The redesigned platform directly targets the practical challenges faced by researchers, emphasizing a more accessible, modular, and adaptable system that can integrate with a variety of existing digital microscopes. This conceptual framework summarizes the key hardware changes and strategies applied to achieve these objectives.

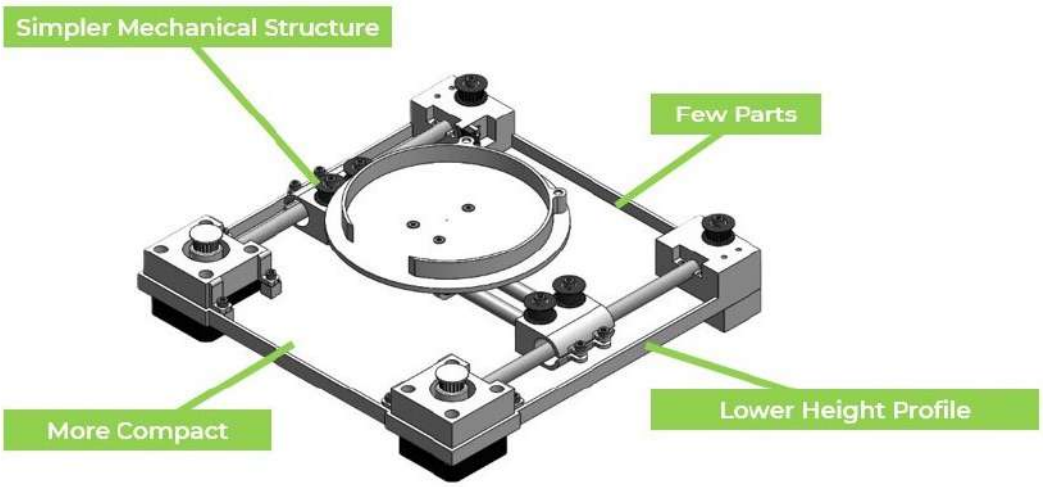


Fig. 4. Conceptual Redesign of the XY Platform V2

The main hardware changes in Version 2 include:

Mechanism redesign

The stacked-gantry mechanism was replaced with a planar H-Bot configuration, significantly reducing height and overall footprint.

Modular subassemblies

Structural components were reorganized into distinct groups, allowing easier assembly, alignment, and maintenance while improving replicability for researchers.

Component optimization for cost

Heavy aluminum extrusions were replaced with linear rods, 3D printing was used more extensively, and other components were sourced from cost-effective suppliers, reducing fabrication costs without compromising precision.

Together, the conceptual redesign and modular subassemblies illustrate how Version 2 directly addresses the key limitations of Version 1. The compact H-Bot configuration reduces the physical footprint, modular grouping simplifies assembly and replication, and optimized components lower fabrication cost. These design decisions not only make the platform more accessible to researchers but also ensure that it can be easily integrated with a variety of existing digital microscopes. The following implementation sections detail how

these conceptual changes were realized in the physical hardware, including CAD designs, measurements, and assembly procedures.

3.3 Development

The second iteration of C.Scope.AI was designed with replicability and accessibility as the central goals. Recognizing that desktop 3D printers are becoming increasingly common in research laboratories, the platform was sized to fit within the build volume of widely available printers, removing the need for industrial fabrication. At the same time, the system maintains precise XY motion for standard Petri dishes and allows integration with a variety of digital microscopes. The following sections describe the design reasoning, component selection, mechanical and electronic integration, and firmware adaptations that together realize a compact, functional, and reproducible Version 2 system.

3.3.1 Decision Making

Target Workspace and Required Movement

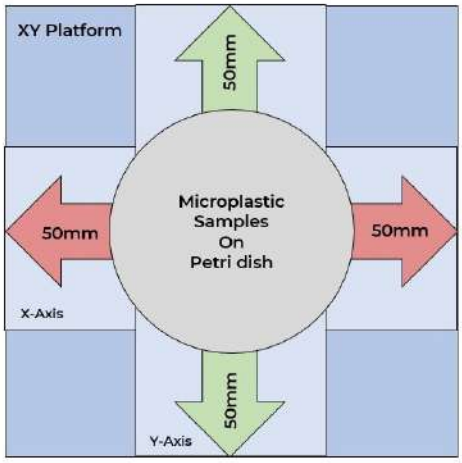


Fig. 5. Required Range of Motion Diagram

The first step in designing Version 2 was defining the functional XY workspace. A 50 mm range of movement in all directions was identified as necessary to cover a standard Petri dish. By clearly establishing these movement requirements, the design was able to maximize reductions in platform size, ensuring that every component was sized and positioned strictly

according to functional needs. This approach produced a compact and adaptable platform suitable for a variety of microscopes.

Overall Platform Size

Fig. 6. Print Bed Boundary as the Platform's Maximum Footprint

Another design consideration was to set the platform's maximum footprint to 220 mm, as most 3D printers have a print bed of this size. This decision allows researchers to fabricate the platform entirely using accessible printers, without specialized industrial equipment. Limiting the size in this way also improves the platform's compatibility with different laboratory setups while maintaining the required XY travel.

Mechanism Selection

Selecting the appropriate XY mechanism was a critical step in designing Version 2 of C.Scope.AI. The goal was to achieve precise XY motion while keeping the platform compact, replicable using standard 3D printers, and adaptable to different microscope setups. Several candidate mechanisms were considered, each offering distinct advantages and limitations:

1. Stacked Gantry: Provided independent X and Y motion with excellent rigidity and minimal backlash. However, it added significant bulk and complexity, limiting compatibility and making replication difficult.

2. Rack and Pinion: Offered moderate resolution and the ability to move heavier loads. Gear meshing introduced mechanical play, vibration, and noise, and alignment complexity reduced replicability.

3. Leadscrew Gantry: Achieved high resolution and minimal backlash with smooth, vibration-free motion. It is structurally rigid but slower, heavier, and requires precision machining, reducing accessibility for 3D printing.

4. Compliant Mechanisms: Utilized flexible material deformation to transmit motion, reducing part count and cost. Precision and rigidity are limited, making them less suitable for reliable micro-scale imaging.

5. H-Bot Planar Mechanism: Uses a single belt in an "H" pattern for coordinated XY motion. It is flat, compact, easier to 3D print and assemble, and offers good positional accuracy, though single-belt actuation can introduce minor racking forces.

6. CoreXY: Crossed belts improve backlash performance and smoothness compared to H-Bot. While slightly more complex, it remains printable on standard 3D printers and supports adaptable setups.

The key performance metrics of each mechanism are summarized in Table II.

Mechanism	**Resolution**	**Backlash**	**Vibration & Damping**	**Structural Rigidity**
Stacked Gantry	High – Precise motion, small step size	Low – Minimal play	Excellent – Smooth, minimal vibrations	High – Supports heavy loads
Rack and Pinion	Medium-Low – Gear teeth limit resolution	High – Gear meshing introduces play	Poor – Gear contact causes vibrations and noise	High – Rigid but may flex if unsupported
Leadscrew Gantry	High – Precise motion, small step size	Low – Tight nut-screw fit	Excellent – Smooth motion	High – Rigid and stable
Compliant Mechanism	Medium – Limited by material deformation	Medium – Elastic deformation introduces play	Low-Medium – Material flexibility may cause oscillations	Low-Medium – Limited by material flexibility
H-Bot	High – Belt stretch may affect accuracy	Medium-High – Single belt racking	Low-Medium – Single belt twisting	Low-Medium – Single-belt frame may twist
CoreXY	High – Improved over H-Bot due to crossed belts	Low – Crossed belts reduce backlash	Good – Smooth belt motion, reduced racking	Medium-High – Depends on frame stiffness

Table II. Comparison of XY Table Mechanisms

Selection Rationale: Considering footprint, replicability, and performance, the H-Bot planar mechanism was selected for Version 2. It was chosen primarily because it provides a

good balance of precision, compactness, and simplicity. It requires fewer non-3D-printed parts and allows modular assembly for future modifications. CoreXY remains a promising alternative for future upgrades where reduced backlash is critical, but for the current iteration, the increased mechanical complexity was not justified.

Component and Material Optimization

Key components were selected based on a balance between functional requirements and cost-effectiveness:

1. Motor Selection: While retaining the same functionality, a smaller variant of the NEMA 17 stepper motors was chosen for Version 2. This decision ensures that performance remains consistent with the previous design while significantly reducing the overall height of the assembly. The compact size reduces the vertical spacing requirement, enhancing integration with various setups.

2. Pulleys and Belts: The platform utilizes GT2 timing belts with 20T GT2 pulleys. This configuration provides smooth motion and efficient power transmission, which is essential for maintaining accuracy during operation. GT2 belts are known for their reliability in various applications, making them an ideal choice for this design.

3. Linear Guides: For the H-Bot mechanism, 8 mm steel rods were chosen as the guide rails. This choice effectively minimized bulk and cost while ensuring ample rigidity, unlike the previous extrusions that contributed significantly to the overall size. Steel rods are known for their durability and stiffness, which are crucial in micro-movement applications.

4. Bearing Selection: To further enhance performance, LM8UU linear bearings were used along the steel rods. These bearings are designed to support smooth linear motion, which is essential for the H-Bot system's operation. Their compatibility with 8 mm rods ensures minimal friction.

5. 3D Printed Components: A significant portion of the platform's structural elements was designed to be 3D printed. This decision not only reduces fabrication costs but also allows for rapid prototyping and modifications based on user feedback. The ability to customize parts easily plays a vital role in enhancing the overall design and user experience.

3.3.2 Design Iterations and Refinement

Building on the design constraints and component considerations outlined previously, including the required XY workspace, platform footprint, and candidate mechanisms, a series of design iterations was explored to address the limitations of Version 1. After selecting the H-Bot planar mechanism as the foundation for Version 2, these iterations focused on refining structural layout, component placement, belt routing, and modular assembly. Each refinement aimed to optimize performance, compactness, and compatibility with digital microscopes, ultimately producing the final Version 2 platform, which balances precision, accessibility, and ease of fabrication.

Iteration V1.0 – Rack-and-Pinion (Historical Attempt)

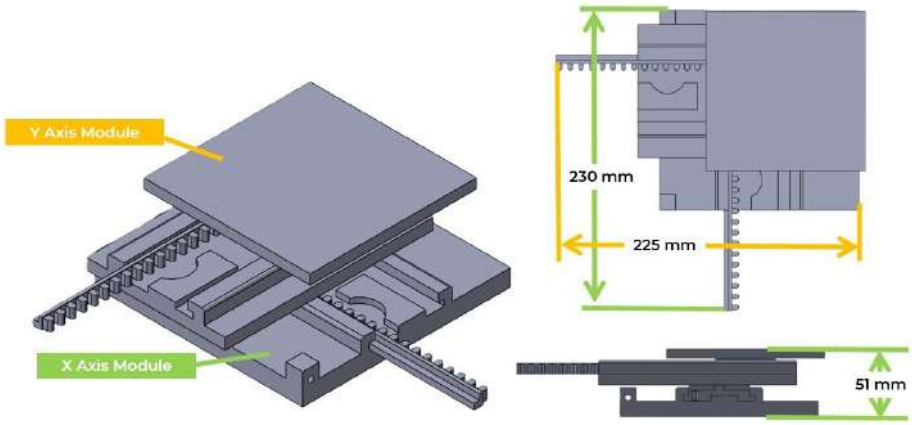


Fig. 7. Early Rack-and-Pinion Prototype

The first iteration explored a rack-and-pinion layout as an initial attempt to achieve a significant reduction in platform size. This 3D model prototype demonstrated a substantial decrease in overall footprint compared to Version 1 and enabled the conceptualization of basic XY motion.

Annotated figures highlight the X-module, Y-module, and the resulting height and width dimensions, which demonstrated that a substantial decrease in physical size was achievable even with a simplified layout. This confirmed that the oversized footprint of Version 1 could be addressed through redesign.

However, it was not pursued further due to several limitations:

1. Backlash and mechanical play from gear meshing reduced positional accuracy.
2. Cantilevered components introduced structural instability.

Although ultimately discontinued, this design served an important role in the development process. It validated that significant size reduction was possible and provided early insights that guided the shift toward a flatter, more compact planar mechanism in later iterations.

Iteration V2.0 – Initial H-Bot Prototype

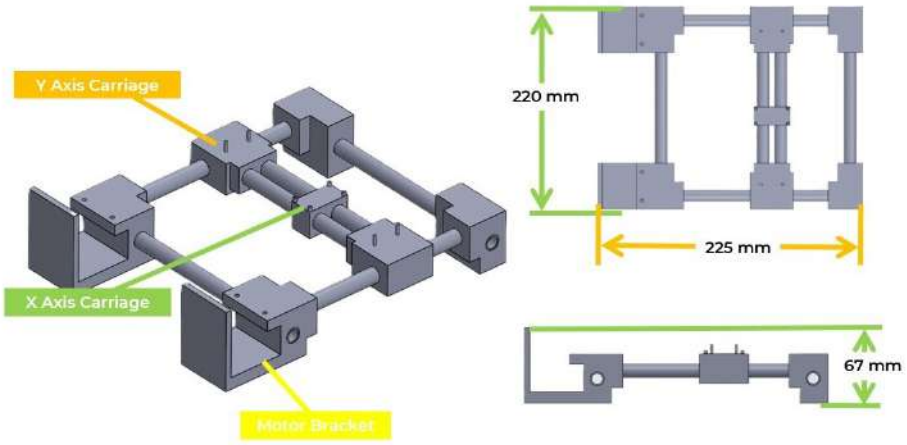


Fig. 8. First H-Bot prototype

Following the limitations of the rack-and-pinion attempt, the second iteration implemented the H-Bot planar mechanism as the basis for a more compact and replicable platform. This 3D-model prototype focused on establishing a sturdier structural layout, as rigidity was identified as the first major requirement for reliable XY motion. The design direction at this stage was to evaluate whether an H-Bot system could realistically fit within the 220 mm × 220 mm printer bed constraint while maintaining mechanical stability.

While the shift to an H-Bot configuration improved layout simplicity and eliminated the backlash issues seen earlier, this iteration introduced several new challenges that prevented it from meeting the size requirement. The prototype became larger than intended due to:

1. Overly bulky, block-shaped bracket assemblies that expanded the frame footprint.
2. Oversized 12.5 mm linear guides, which added unnecessary mass and increased the system's overall profile height.

3. Clearance requirements for motors and belt routing that pushed the layout beyond the 220 mm × 220 mm limit.

Although this iteration successfully established the long-term architectural direction, its excessive bulk and over-engineered components emphasized the need for slimmer brackets, lighter linear motion elements, and a more space-efficient structural arrangement. These insights directly guided the refinements pursued in the next iteration.

Iteration V2.1 – Slimmed H-Bot Prototype

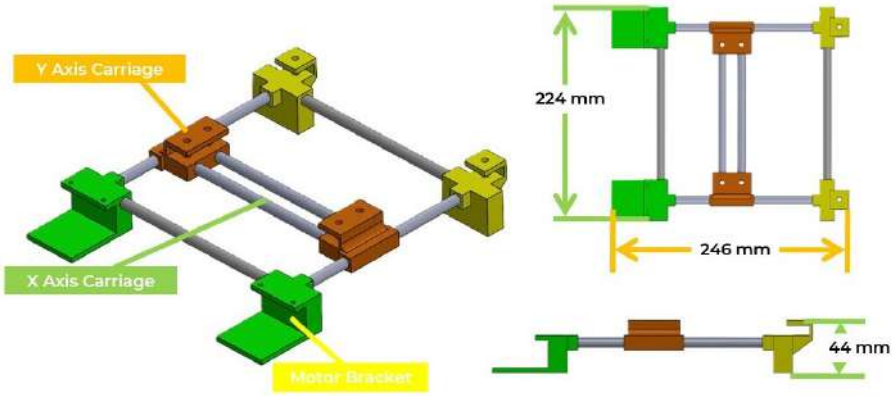


Fig. 9. Second H-Bot prototype

The third iteration focused on reducing overall mass and lowering the platform's profile. The oversized 12.5 mm linear guides from the previous prototype were replaced with 8 mm guide rails, decreasing weight and slightly reducing the system's height. Bracket assemblies were also minimized to make the structure leaner and improve compatibility with standard 3D printing constraints.

Despite these improvements, several challenges remained:

1. The platform still did not fit within the 220 mm × 220 mm 3D-printer bed, as clearance for motors and belts continued to expand the footprint.
2. The reduced 5 mm rods, while lighter, raised concerns about rigidity, potentially affecting precise XY motion.
3. Bracket designs became overly specific and intricate, requiring additional support material during printing, increasing print time, material use, and the risk of print failure.

4. Motor mounting lacked sufficient strength, raising concerns about stability and vibration resistance during operation.

Overall, this iteration successfully reduced bulk and refined proportions, but the combination of limited rigidity, impractical bracket geometry, and insufficient motor support highlighted the need for a more robust, manufacturable design in the subsequent iteration.

Iteration V2.2 - Simplified H-Bot Prototype with Reduced Rod Diameter

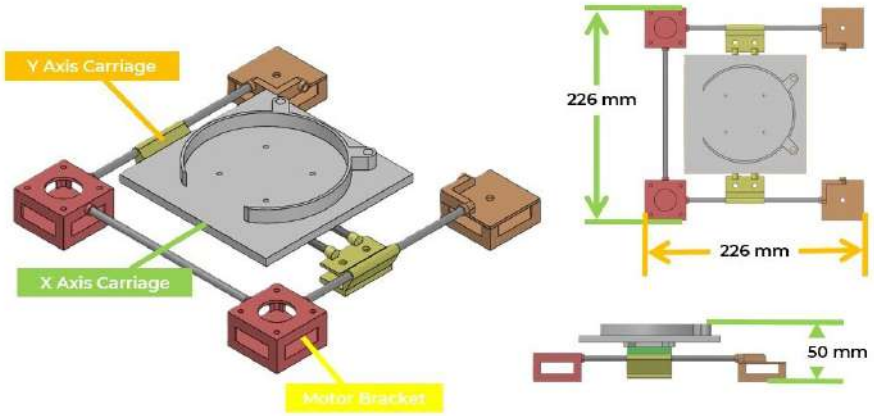


Fig. 10. Third H-Bot prototype

The fourth iteration focused on improving manufacturability and simplifying the mechanical structure while further reducing the platform size. Bracket designs were made less intricate and more 3D-print-friendly, and the linear guide rods were increased back to 8 mm to provide better rigidity compared to the previous slimmed-down version. These changes lowered print complexity and enhanced structural stability.

However, several challenges persisted:

1. The platform still could not fit within the 220 mm × 220 mm 3D-printer bed, as the overall footprint remained constrained by motor and belt placement.
2. Alignment of components during assembly posed potential issues, as individual brackets were still separate pieces, leaving tolerance stacking and misalignment risks.
3. Although brackets were simplified, careful assembly was still required to maintain XY precision, highlighting the need for a unified frame in the next iteration.

Overall, this iteration succeeded in reducing print complexity, improving bracket design, and enhancing rod stiffness, but it revealed that a more integrated structural solution would be necessary to achieve consistent alignment and the desired compact footprint.

Iteration V2.3 - Unified Four-Corner Frame

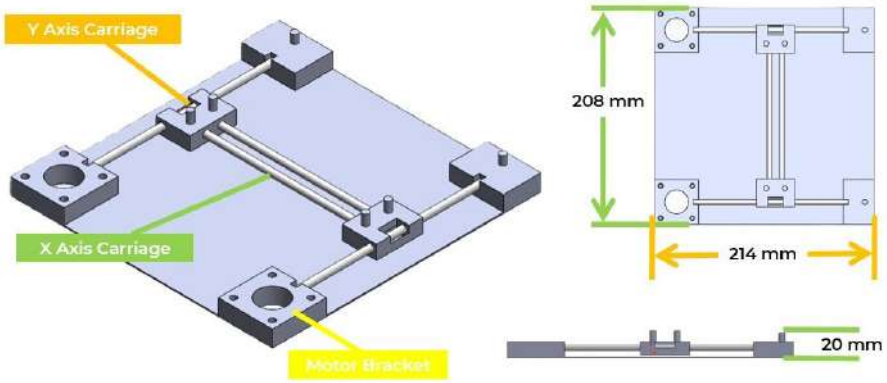


Fig. 11. Fourth H-Bot prototype

The fifth iteration introduced a major structural refinement by consolidating all brackets into a single continuous base frame. By integrating the four corners into one solid, 3D-printable plane, alignment issues were eliminated, and printing complexity was significantly reduced. This design also allowed the motors to be mounted underneath the platform rather than alongside it, freeing up space and enabling the system to fit comfortably within the 220 mm × 220 mm 3D-printer bed.

Minor adjustments were introduced to further optimize performance:

1. Outline ribs were added to the frame to prevent flexing during operation and maintain XY precision.
2. Motor insertion slots were incorporated, allowing the motor pulleys to be raised to the correct level and aligned with the other pulleys, ensuring smooth belt tensioning.

Overall, this iteration produced a compact, rigid, and easily manufacturable H-Bot platform, addressing the key limitations of previous prototypes while laying the foundation for the final refined version.

Final Iteration - Refined H-Bot Platform

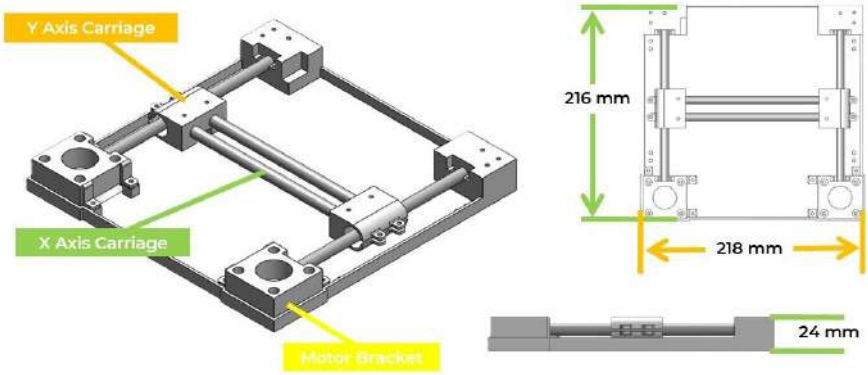


Fig. 12. Final H-Bot prototype

The final iteration built upon the unified four-corner frame, introducing the last set of mechanical refinements to produce the platform used in the current system. Motor brackets were redesigned as separate, attachable components that interface with the main frame, allowing precise positioning and easy maintenance. Motor slots were incorporated to ensure pulley alignment with the belts, improving motion consistency and tensioning. Outer ribs were added around the platform to enhance rigidity and reduce flex during rapid traversal. Rod retention features with click-fit designs simplified assembly and maintenance.

This iteration represents the culmination of the iterative design process and serves as the foundation for the Version 2 platform used in C.Scope.AI.

Final Platform Design

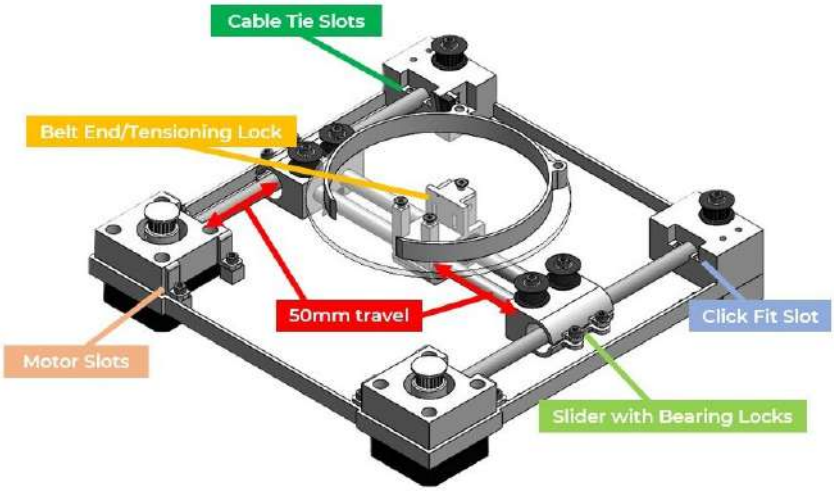


Fig. 13. 3D Rendered Final Prototype

Building on the final iteration, the Version 2 platform was first finalized as a detailed 3D model. A previous section (Solution) introduced this model, illustrating the overall layout and design. In this section, the model is presented with annotated features, highlighting the mechanical improvements developed throughout the iterative process, and is followed by the physical assembly of the prototype. This allows a clear comparison between design intent and realized hardware, demonstrating the implementation of key features in the constructed system.

Key features included in the platform are as follows:

1. XY Sliders: LM8UU bearings secured in slots with locking features to ensure smooth and accurate XY motion.
2. Belt-locking mechanism and adjustable tensioning system to maintain proper belt tension and prevent slippage.
3. Motor Brackets and Slots: Motors mounted in dedicated slots beneath the platform, aligning pulleys with the belts and minimizing footprint.
4. Outer Ribs and Unified Frame: Enhance rigidity and reduce flex during rapid motion.

5. Rod Retention / Click-Fit Features: Facilitate easy assembly, maintenance, and replacement of rods.

3.4 Mechanical Integration

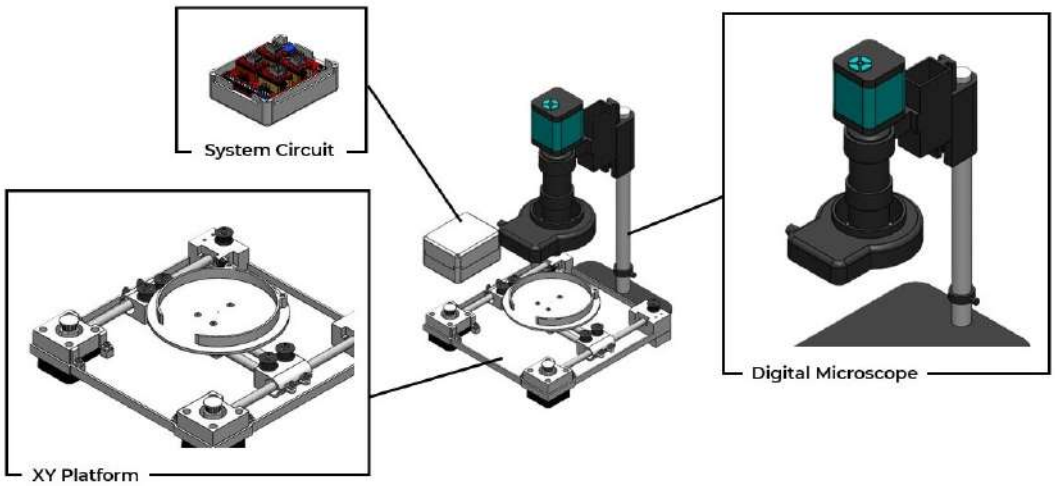


Fig. 14. Hardware Overview

Following the finalization of the H-Bot platform design, the complete Version 2 hardware assembly of C.Scope.AI was planned and integrated. While the platform serves as the mechanical foundation, the system also includes the microscope, limit switches, motors, belt transmission, and control electronics. This section describes how these components were positioned, mounted, and interconnected to form the fully operational hardware, ensuring that all subsystems work together cohesively to achieve precise XY motion and reliable imaging performance.

3.4.1 Digital Microscope

Fig. 15. H1600 Digital Microscope

C.Scope.AI V2 uses the H1600 digital microscope as the default optical device for image acquisition. The microscope was positioned above the XY platform according to its required working distance and field-of-view constraints.

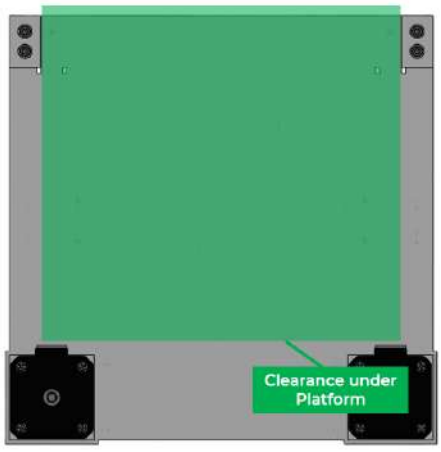


Fig. 16. Spatial Clearance for Digital Microscopes

The platform provides an internal clearance of approximately 180 mm in width, 165 mm in length beneath the base, and up to 13.5 mm in height. To support different microscope geometries during system assembly, an external stand may be attached when positioning alternate microscope models with bases higher than 13.5 mm.

3.4.2 Transmission

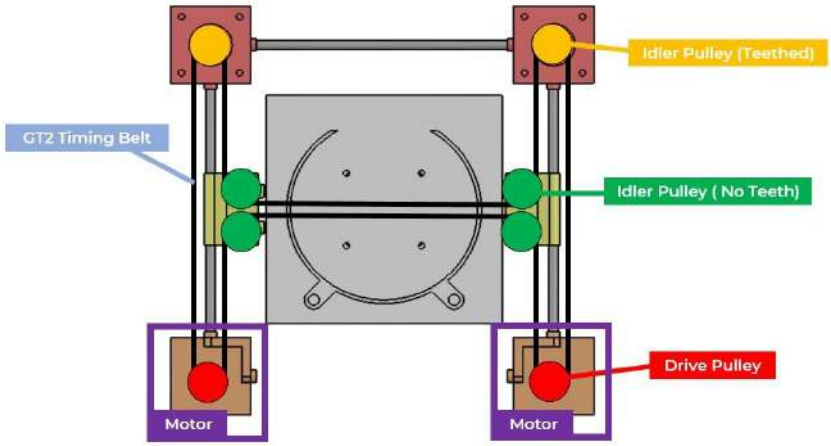


Fig. 17. H-Bot Configuration

The Version 2 platform employs an H-Bot transmission system. A GT2 timing belt is routed around a set of pulleys configured to form an "H" pattern, and two stationary NEMA 17 stepper motors on the same level drive the belt from fixed locations on the frame. Belt tension is adjusted using an integrated tension lock on the platform's central carriage, where both belt ends may be secured. Coordinated motor actuation produces translation across the XY plane according to the H-Bot motion sequence.

3.4.3 Limit Switches

Micro-limit switches are installed to define positional reference and end-stop protection for each axis. These switches are mounted at designated end-stop locations on the frame and connected to the CNC control circuit of the system. They provide homing signals to the firmware and function as boundary sensors for the motion system.

3.4.4 XY Assembly

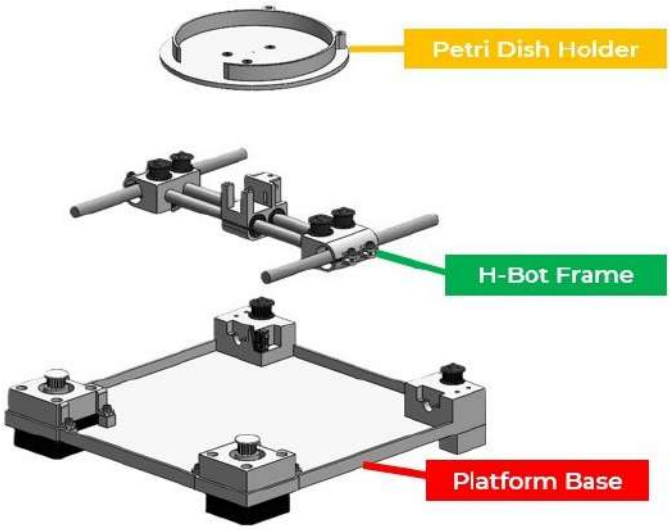


Fig. 18. XY Platform V2 Mechanical Composition

The XY table in C.Scope.AI V2 is organized into three primary mechanical groups: the platform base, the H-Bot frame, and the Petri dish holder.

Platform Base

The platform base serves as the main structural foundation of the XY system. It supports all upper components, directly holds both NEMA 17 motors, and houses the mounting points for the linear motion system.

H-Bot Frame

The H-Bot frame sits on top of the main base. It contains the linear guides, sliders, pulleys, and central carriage. These components were assembled to form the "H" frame and motion structure. The frame includes the belt end locks and attachment bracket for the Petri dish holder.

Petri Dish Holder

The Petri dish holder attaches to the central carriage of the H-Bot frame and secures the sample container during imaging. It is designed for 100 mm diameter Petri dishes, with optional modification for alternative dish sizes. The holder is positioned to align with the microscope's field of view and move consistently with the XY motion.

3.4.5 Control System

The control electronics consist of an Arduino Uno paired with its CNC shield and DRV8825 motor drivers for both X and Y axes. These components were housed in a single casing located beside the XY platform and wired to the motors, limit switches, and power input following the existing control schematic. A separate casing was made to protect the circuitry and allow minimal, organized wiring.

3.4.6 Firmware

C.Scope.AI V2 continues to utilize GRBL firmware as the primary control and motion-planning engine. GRBL interprets G-code commands and generates precise control signals for the motor drivers. Importantly, the transition from a stacked-gantry architecture to an H-Bot mechanism only required minimal targeted modifications to the GRBL firmware to support H-Bot kinematics. Specifically, the motor-mapping configuration was adjusted so that each commanded X or Y movement generates the appropriate mixed step outputs across both motors. Beyond this kinematic remapping, the firmware's planning algorithms, homing procedures, and overall operational structure remain unchanged.

3.5 Final Version 2 Prototype

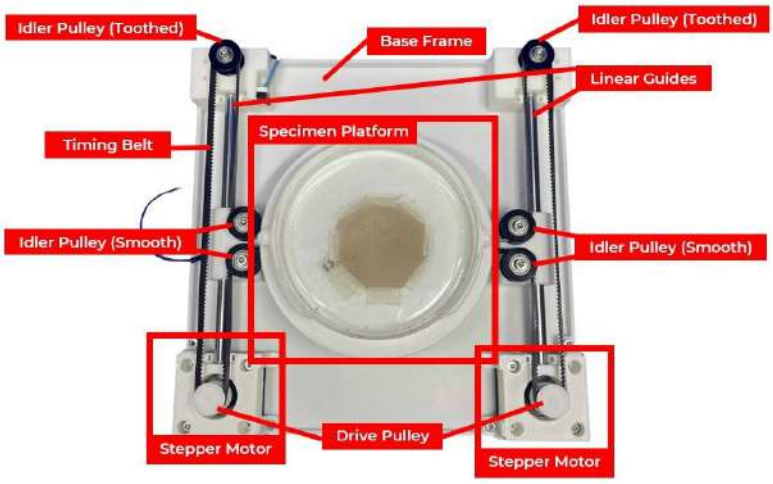


Fig. 19. XY Platform Version 2

With all mechanical components, motor mounts, linear guides, belts, and control interfaces fully planned and integrated, the assembled Version 2 platform was built to verify manufacturability, rigidity, and overall functionality. Figure 19 presents the real prototype with annotations, confirming that the design features developed through the iterative process were successfully implemented in the physical system. Together, these figures provide a complete overview of the final platform and demonstrate the realization of the fully integrated C.Scope.AI hardware.

3.5.1 Tuning

With the hardware of the Version 2 H-Bot platform finalized, the system's mechanical performance was fine-tuned to achieve its full potential. In particular, belt tension directly affects positional accuracy, repeatability, and overall imaging quality. Incorrect tension can introduce alignment errors, reduce XY precision, and compromise the reliability of microplastic imaging.

Therefore, a belt tension calibration procedure was performed to determine the optimal setting that ensures consistent and accurate motion.

Calibration Procedure

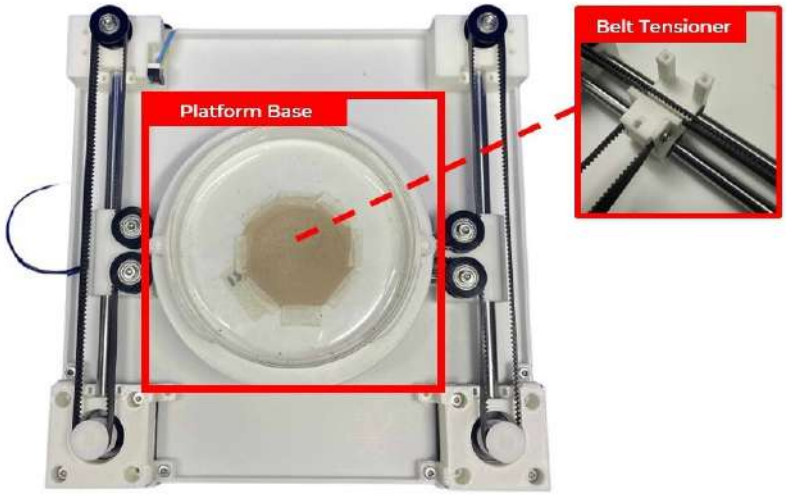


Fig. 20. Belt End Locks / Tensioning Mechanism

1. The platform base was temporarily removed to access the belt tensioners located beneath the H-Bot frame.

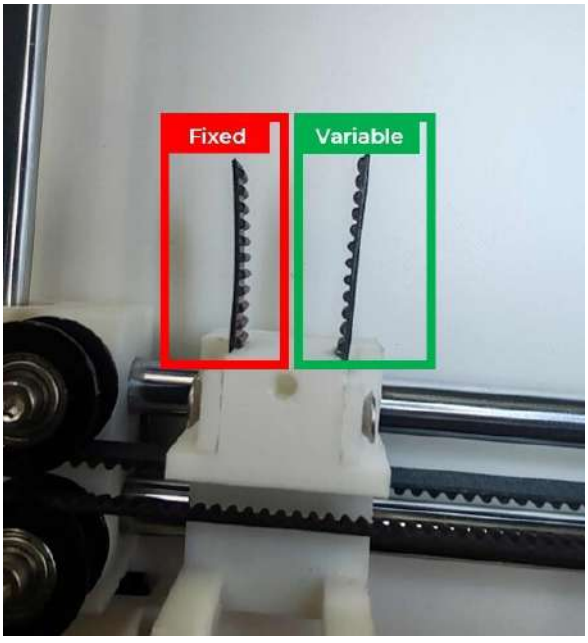


Fig. 21. Fixed and variable belt ends for belt tension adjustment

2. Initial belt exposure was set to 11 teeth on the fixed end and 12 teeth on the variable end.

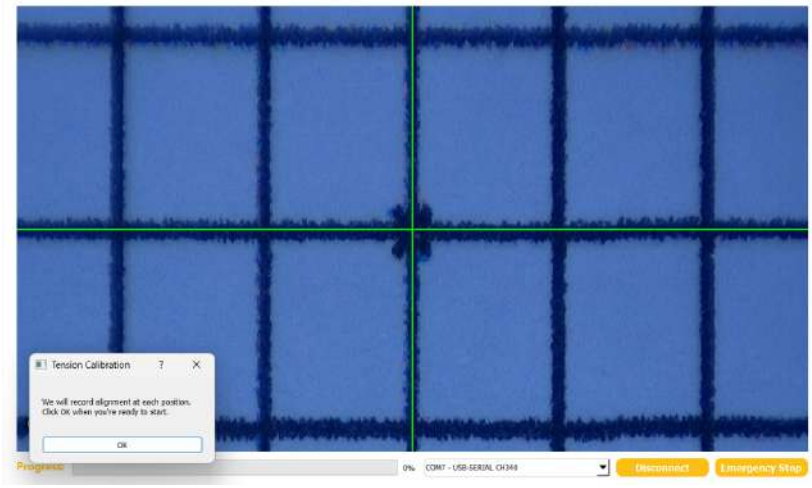


Fig. 22. Grid Alignment

3. A calibration grid was placed on the platform, and the platform was homed to align the grid center with the software crosshair.

4. Belt tension on the variable end was adjusted incrementally across a range of settings, from very tight to very loose.

5. After each adjustment, the platform base was reattached, and XY positioning accuracy was measured using the calibration routine.

6. X and Y positional errors were recorded for each tension setting to evaluate performance.

Performance Impact

The findings below demonstrate the effect of varying belt tensions on platform alignment:

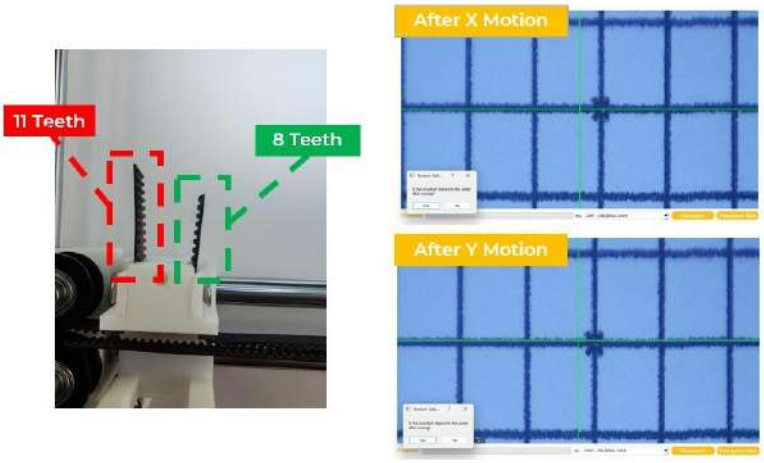


Fig. 23. Loose (8 teeth exposed) Tension Results

1. Loose Tension: Slack in the belt produced noticeable misalignment along both axes, degrading imaging precision.

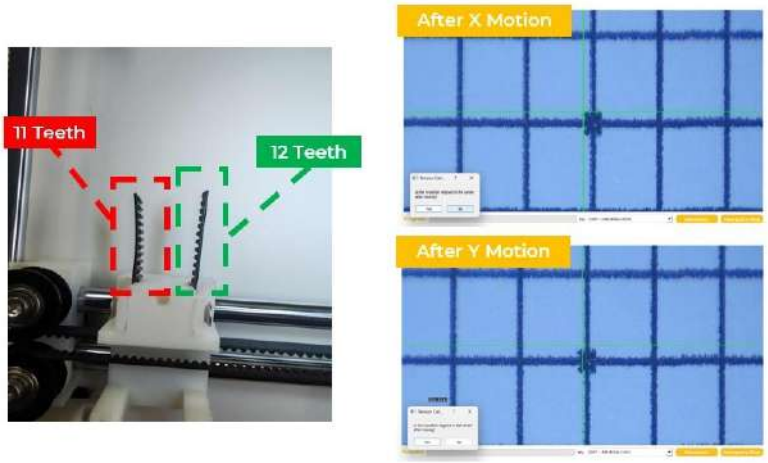


Fig. 24. Tight (12 teeth exposed) Tension Results

2. Tight Tension: Excessive tension introduced friction and racking forces, causing uneven motion and potential stress on mechanical components.

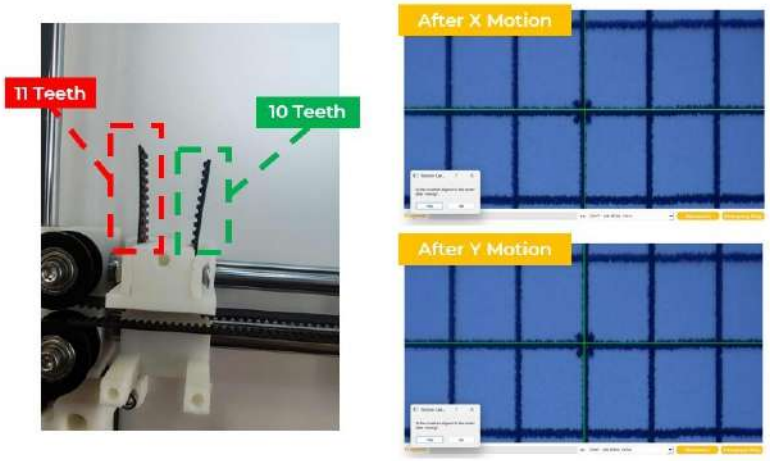


Fig. 25. Optimal (10 teeth exposed) Tension Results

3. Optimal Tension: A balanced tension minimized positional errors and ensured smooth, repeatable XY motion, directly improving imaging accuracy and reliability.

The optimal belt tension identified through this procedure was adopted as the standard for all system operations, ensuring that the H-Bot platform maintains high positional accuracy and consistent performance during microplastic imaging tasks.

3.6 Results and Discussion

The redesigned hardware focused on enhancing compactness, adaptability, cost efficiency, and ease of assembly and maintenance. The primary objectives guiding these efforts were the development of an automated XY platform capable of interfacing with a range of existing microscopes and reducing overall hardware costs while preserving system functionality. Key electronic components were retained, including stepper motors, motor drivers, and the microcontroller, while the mechanical layout was updated to a streamlined H-Bot configuration.

3.6.1 Size Reduction

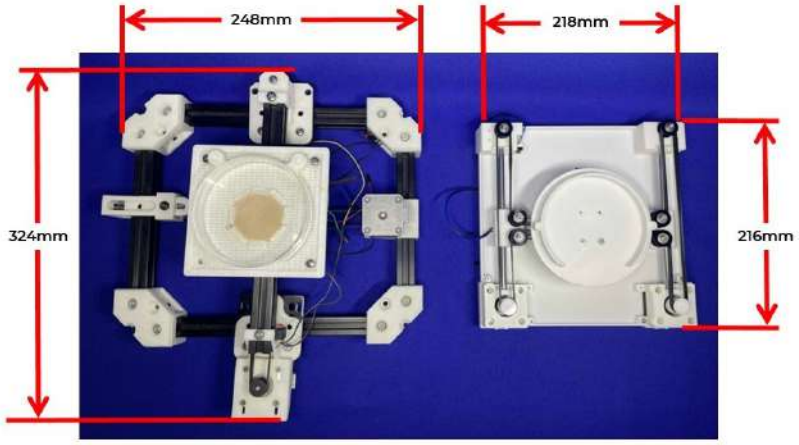


Fig. 26. Footprint comparison between Version 1 and Version 2

Fig. 27. Height comparison between Version 1 and Version 2

A major outcome of the redesign is the reduction in the platform's dimensions. Version 2 achieves a significant 42% reduction in footprint area and a 55% reduction in height profile compared to Version 1. The reduced lateral dimensions and lower profile allow the platform to fit between the microscope objective and base while holding a Petri dish, without interfering with imaging operations. This lower height also facilitates positioning beneath microscopes with limited vertical clearance. These changes support hardware objectives by achieving a compact and space-efficient design suitable for multiple laboratory setups.

3.6.2 Adaptability

The smaller and flatter design increases adaptability. The platform's dimensions allow integration with the geometric constraints of various digital microscopes. In practice, the platform can be positioned directly under the microscope or mounted using an external support stand when additional clearance is required. During testing, the platform was positioned directly under the default H1600 microscope as well as two additional microscope

models to verify clearance and positioning. For setups requiring extra vertical space, the platform was mounted using an external support stand, such as when the microscope base exceeded the default 13.5 mm clearance. An adjustable leadscrew stand was used to confirm alignment and motion across all tested setups. These results demonstrate that the redesigned XY platform can function with multiple microscope geometries, fulfilling the hardware objective of compatibility across existing laboratory systems.

3.6.3 Cost Reduction

The transition from the stacked-gantry configuration in Version 1 to the H-Bot layout in Version 2 contributed to a reduction in both part count and material requirements. The total hardware cost decreased from PHP 14,530 in Version 1 to PHP 8,934 in Version 2, representing a 23% reduction. The simplified mechanical design replaced bulky aluminum extrusions with a combination of 3D-printed components and a compact H-Bot frame, reducing fabrication complexity. These design choices support the hardware objective of enhancing cost-effectiveness while maintaining operational functionality.

3.6.4 Maintenance and Assembly Simplicity

The redesign also simplified assembly and maintenance. The modular organization of the XY table into the platform base, H-Bot frame, and Petri dish holder reduces the number of unique parts and consolidates 3D-printed modules. Components are more accessible for servicing and alignment without extensive disassembly. These features support the objective of maintaining a user-friendly and serviceable system.

3.6.5 Performance

Following calibration of the belt tension, the XY platform's positional accuracy and repeatability were evaluated. With the optimal tension applied, the platform was subjected to five consecutive test runs across the same predefined XY motion sequence. A calibration grid was used to measure alignment at key reference points for each run.

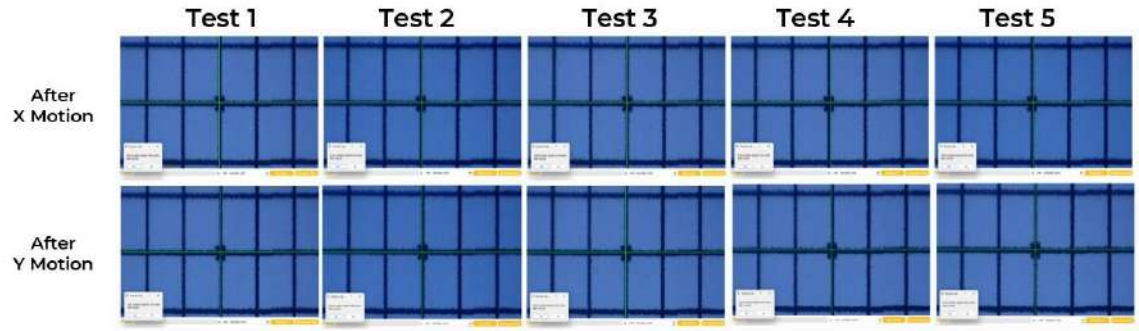


Fig. 28. Platform Movement Repeatability Results

The results demonstrate consistent positioning across all five tests, with minimal deviation in both X and Y axes. This confirms that the calibrated belt tension ensures smooth, accurate, and repeatable motion, supporting reliable imaging of samples. The high repeatability achieved validates that the system can consistently perform microplastic scanning tasks without misalignment or drift, fulfilling the intended performance objectives of the Version 2 platform.

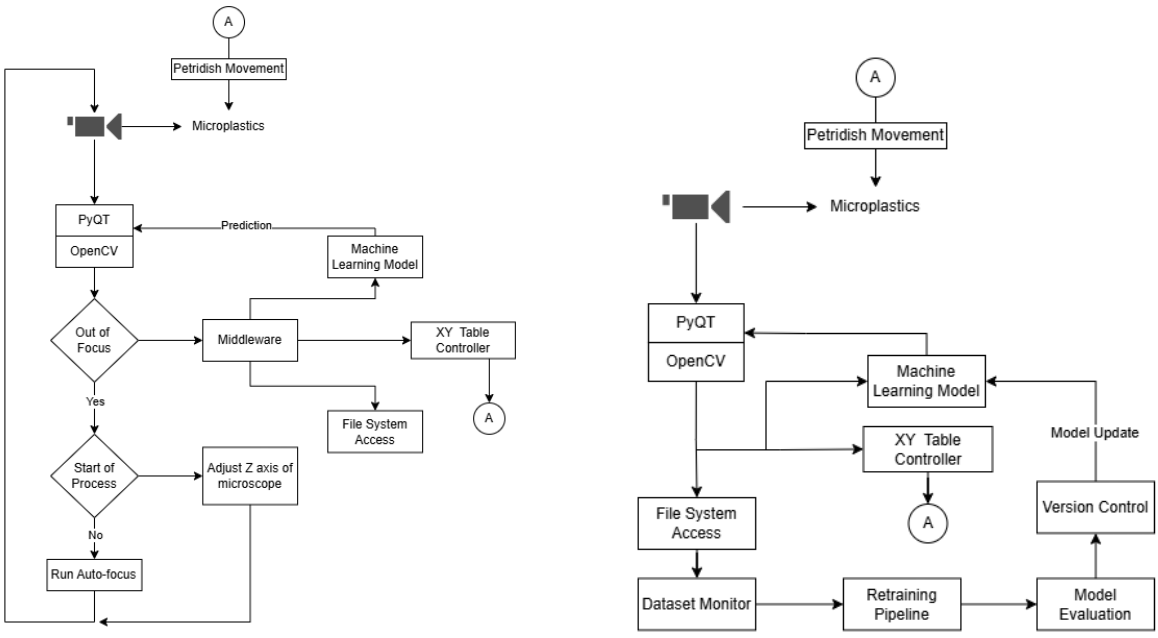
Chapter 4

Software Architecture

This section outlines the software re-engineering process implemented to optimize the system's architecture. Moreover, results and discussions are presented to validate the performance enhancements through architectural redesign.

4.1 Methodology

4.1.1 Software Architecture Development



(a) Software Architecture V1 **(b)** Software Architecture V2

Fig. 29. Software Architecture Comparison Between Versions

A. Software Architecture V1

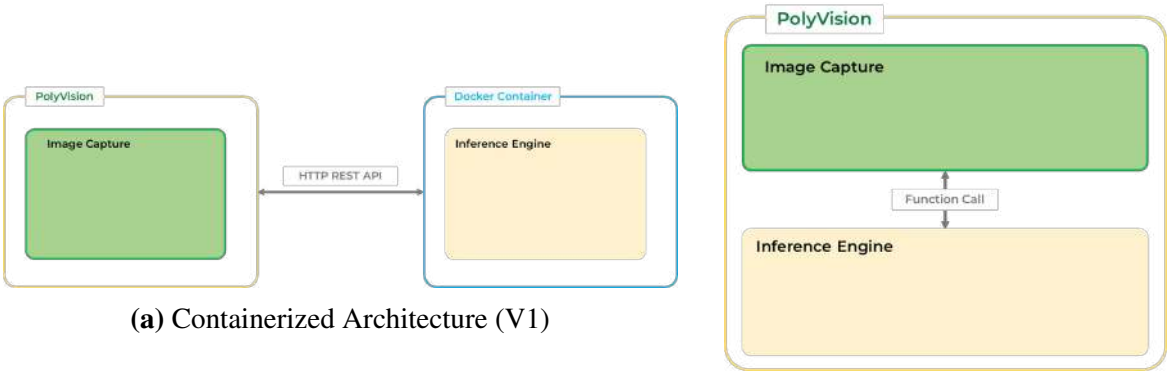
The initial software architecture functioned as a distributed system where a **Docker Container** serves as the *middleware* to orchestrate detection operations. As illustrated in Fig. 29a, the middleware serves as an intermediary for all communications between PyQT/OpenCV and the machine learning model. This design necessitated a containerized approach where the model operated in isolation, effectively acting as a separate server that required data serialization to pass image frames via an HTTP REST API. The architecture was primarily static and focused solely on the operational detection loop, capturing images, checking focus, and classifying samples, without any integrated mechanism for model improvement or data management. Furthermore, the serialization required by the detection process introduced significant system latency.

B. Software Architecture V2

In Version 2, the architecture was re-engineered into a localized, integrated application to enhance performance and capability. As illustrated in Fig. 29b, the middleware layer which is the Docker was eliminated entirely. The PyQT/OpenCV interface was updated to communicate directly with the Machine Learning Model, utilizing shared memory and direct function calls rather than network-based requests. Furthermore, Version 2 introduced a closed-loop Retraining Pipeline that is not implemented in the first iteration. This includes *Dataset Monitor* to flag new samples, a *Retraining Pipeline* for local model updates, and a *Version Control* system to manage model deployments. This architectural expansion transformed the software from a static detection tool into an adaptive system capable of continuous, autonomous learning.

4.1.2 Inference Architecture

To address the latency and reliability issues identified in the first iteration, the software underwent a fundamental redesign, transitioning from a Containerized Microservice Architecture to a Monolithic Integrated Architecture.



(a) Containerized Architecture (V1)

(b) Localized Architecture (V2)

Fig. 30. Comparison of Inference Architectures

A. Containerized Inference (Version 1)

The initial system relied on Docker to host the machine learning model within an isolated container. As illustrated in Fig. 30a, this required the main application to communicate with the model via HTTP requests (REST API). While modular, this introduced two critical bottlenecks:

- **Serialization Overhead**. Every image frame captured had to be serialized (encoded to JSON/Base64), transmitted over a virtual network port, and deserialized by the container before inference could begin.
- **Resource Partitioning**. The Docker engine (specifically the Vmmem process on Windows) reserved a fixed block of RAM regardless of the model's actual usage, causing high memory strain on lower-end research laptops.

B. Localized Integrated Inference (Version 2)

Version 2 architecture removed the virtualization layer entirely. As shown in Fig. 30b, the PyTorch inference engine was embedded directly into the application's runtime environment. The model was loaded into the system's shared memory, which allowed the application to pass raw image tensors directly to the Central Processing Unit (CPU) without serialization. This CPU-centric approach was adopted to ensure compatibility with standard research laptops lacking dedicated Graphics Processing Units (GPUs). This redesign unified the User Interface and the Intelligence Module into a single process, eliminated the dependency on the Docker daemon, and ensured that the system functioned reliably in offline, resource-constrained environments.

4.1.3 Implementation of Localized Architecture

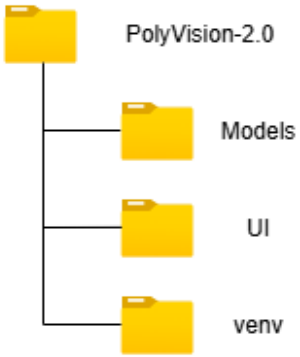


Fig. 31. File Directory of the Project

As seen in Fig. 31, the model/s and UI are in the same root directory (PolyVision-2.0), which provides a seamless localization of the application. The Models folder contains the model of the first iteration that is trained, using the hyperparameters listed in Table III [19]. The model is then extracted from their Docker container to the local file. This ensures that there is no accuracy loss in the models. It is deployed using a .pth format for PyTorch. On the other hand, the UI folder included the graphical user interface developed during the first iteration of the application, functioning as the primary component that facilitates interaction between the system and the end user. A Python 3.10 virtual environment (venv) was established for the development of the application, requiring the installation of specific packages, including PyQT and OpenCV for the user interface and computer vision, and PyTorch and Detectron2 for the machine learning model. The venv module in the Python standard library provides an isolated environment for the project, thereby preventing conflicts with dependencies from other projects. The software interface follows an MVC Architecture which was established by the initial iteration.

Hyperparameter	Value	Classification Type
Max Iterations	10k	Binary (Bin)
	12k	Multiclass (MC)
Base Learning Rate	0.01	Bin and MC
Batch Size	128	Bin
	256	MC
Weight Decay	0.001	Bin and MC
Learning Rate	MultiStepLR	
Warmup iterations	1k	
Warmup Method	Linear	
Warmup Factor	0.001	
Gamma	0.1	
Scheduler Steps	(2k, 5k, 8k)	Bin
	(2k, 5k, 8k, 10k)	MC

Table III. Hyperparameter table of values for binary classification and multi-classification models

4.2 Results and Discussions

This section demonstrates how the software re-engineering process enhanced system accessibility, fostering a more direct and efficient user experience.

4.2.1 Execution Time Comparison

Following successful localization of the system, the research team conducted a quantitative performance assessment of the AI model's inference pipeline—specifically, the time required to generate bounding boxes, assign classifications, and compute confidence scores. Two distinct system configurations were evaluated to compare execution times across platforms. The experiment used 90 images, representing 30% of the full dataset.

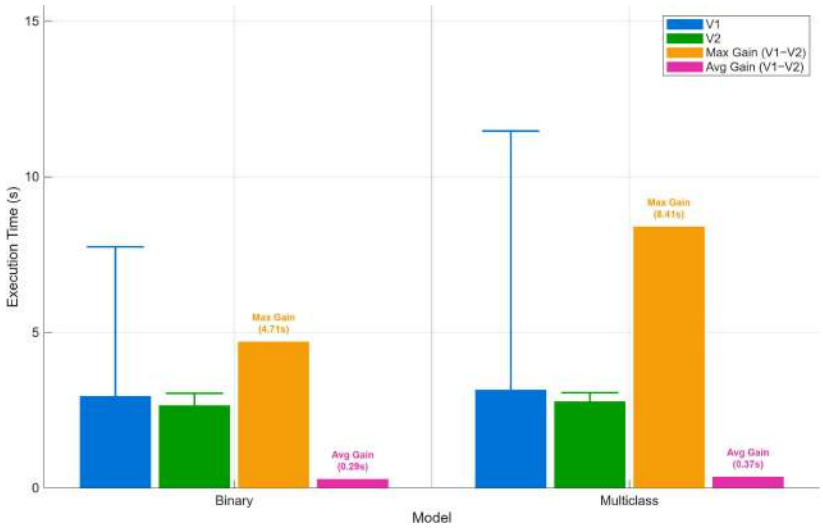


Fig. 32. Execution Time Comparison of the System using Intel(R) Core(TM) i3-1115G4

As illustrated in Fig. 32, Version 2 (V2) demonstrates substantially faster performance than Version 1 (V1) for both the binary and multiclass models. Specifically, the binary model exhibits a 9.80% speedup, while the multiclass model achieves an 11.71% speedup. In terms of worst-case execution time, V2 reduced the maximum time by 4.71 seconds for the binary model and by 8.41 seconds for the multiclass model, indicating that V2 handles peak-load scenarios significantly more efficiently. On average, V2 also completed classification 0.29 seconds faster for binary and 0.37 seconds faster for multiclass, suggesting consistent performance improvement across typical inference runs. Moreover, Table IV presents the maximum time, minimum time, average time, and standard deviation using Intel(R) Core(TM) i3-1115G4.

	Version 1		Version 2	
	Binary	Multiclass	Binary	Multiclass
Maximum Time (s)	7.76	11.48	3.05	3.07
Minimum Time (s)	2.52	2.65	2.42	2.65
Average Time (s)	2.96	3.16	2.67	2.79
Std. Deviation	0.66	1.00	0.14	0.08

Table IV. Summary of the Execution Time using Intel(R) Core(TM) i3-1115G4

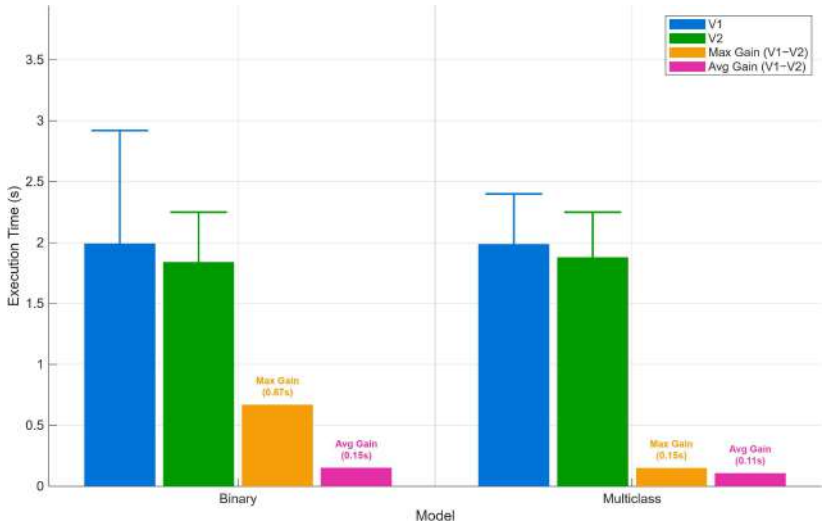


Fig. 33. Execution Time Comparison of the System using Intel(R) Core(TM) i7-10510U

Similarly, as shown in Fig. 33, V2 performed significantly better than V1 in both models. The binary model achieved a 7.54% speedup, while the multiclass model achieved a 5.53% speedup. Regarding worst-case execution time, V2 reduced the maximum time by 0.67 seconds for the binary model and by 0.15 seconds for the multiclass model, reflecting a more controlled upper bound in processing time. On average, V2 completed classification 0.15 seconds faster for the binary model and 0.11 seconds faster for the multiclass model, demonstrating a consistent, albeit modest, improvement in typical inference speed across both model types. Moreover, Table V presents the maximum time, minimum time, average time, and standard deviation using Intel(R) Core(TM) i7-10510U.

	Version 1		Version 2	
	Binary	Multiclass	Binary	Multiclass
Maximum Time (s)	2.92	2.44	2.25	2.33
Minimum Time (s)	1.74	1.73	1.77	1.75
Average Time (s)	1.99	1.99	1.84	1.88
Std. Deviation	0.18	0.13	0.07	0.08

Table V. Summary of the Execution Time using Intel(R) Core(TM) i7-10510U

The execution time is evaluated with two different systems, and the findings indicate a substantial performance improvement resulting from the architectural modifications implemented in Version 2. The reduced and stable processing time across both systems indicates that the architectural redesign is more efficient in managing computational workloads. These

findings further demonstrate that the optimizations in Version 2 enhance overall system responsiveness and operational efficiency, regardless of hardware capability.

4.2.2 Memory Utilization Comparison

(a) Version 1

Process Name	Memory Usage (Avg.)
Docker (Docker App & Models)	103.2 MB
WSL	915.6 MB
Application (PolyVision)	891.4 MB
TOTAL	~1,910.2 MB

(b) Version 2

Process Name	Memory Usage (Avg.)
Application (PolyVision & Models)	450.8 MB
TOTAL	~450.8 MB

Table VI. Memory Utilization Comparison Between Versions

Memory utilization was analyzed to highlight the architectural differences between the two software versions. As presented in Table VI, Version 1 operates within a Docker environment, which contributes additional overhead to its overall memory consumption. A significant portion of this overhead is attributed to the Windows Subsystem for Linux (WSL), which necessitates a dedicated memory allocation to sustain the virtualized Linux kernel and file caching mechanisms required for container orchestration. In contrast, Version 2 runs solely as a local application, utilizing only the system's native memory resources, as the models are stored locally and no longer rely on external dependencies. Version 2 demonstrates lower memory usage compared to Version 1, showing a difference of approximately **1,459.4 MB (1.5 GB)** between the two. This reduction indicates an improvement in the software architecture implemented in Version 2.

4.2.3 System Setup Comparison

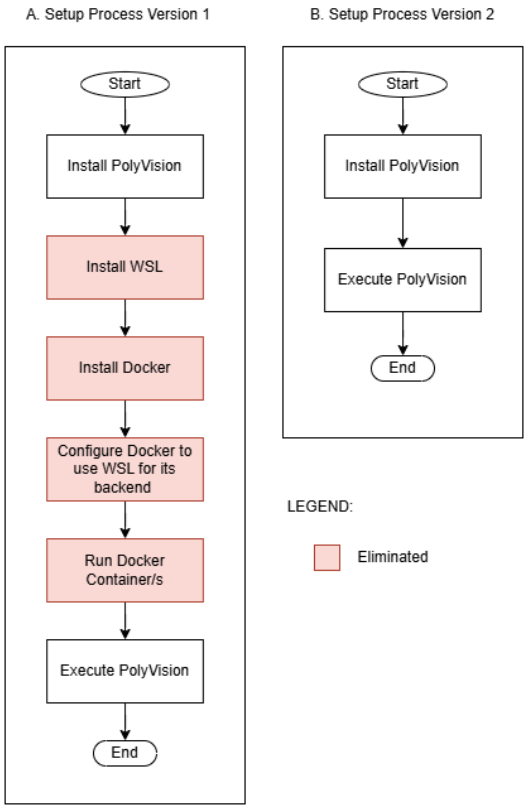


Fig. 34. Setup Process Comparison

Figure 34 illustrated the substantial reduction in system setup and deployment complexity achieved in the second iteration of the software. The setup procedure illustrated in Fig. 34A. was characterized by a multi-stage dependency chain, necessitating the installation and configuration of the Windows Subsystem for Linux (WSL) and the Docker Desktop engine prior to application execution. This architecture required end-users to manually bridge the Docker backend with the WSL environment and run the Docker container before the software application, *PolyVision* interface could become operational. Consequently, this process introduced difficulty, especially those who are non-tech-savvy users.

In contrast, Version 2 streamlines the setup process and deployment workflow by eliminating the virtualization layer entirely. The machine learning model was loaded in the system's shared memory, negating the requirement for external dependencies such as WSL and Docker. This resulted in a linear, two-step "install-and-execute" procedure where the user simply installs the standalone *PolyVision* application and executes it directly. A compre-

hensive *User Guide* is also provided for users to provide seamless operational workflow. This optimization transformed the initial system being a complex, dependency-heavy deployment into a standalone executable, significantly enhancing user accessibility and streamlining the deployment workflow.

4.3 Summary

In summary, the transition from a containerized to a localized architecture yielded significant improvements across performance, resource efficiency, and usability. Factual benchmarks confirmed an inference speedup of up to 11.71% and a memory reduction of approximately 1.5 GB, validating the elimination of middleware overhead. Furthermore, the streamlined deployment process removed complex dependency requirements, transforming the system into an accessible standalone solution suitable for diverse research environments.

Chapter 5

Retraining Feature

This chapter details the development and validation of the local retraining module integrated into C.Scope.AI V2, examining both its operational mechanics and performance metrics. To ensure the feasibility of widespread deployment, a comprehensive suite of machine learning strategies was applied, encompassing hyperparameter optimization, architectural modifications, and dynamic scaling. The resulting pipeline was validated against rigorous benchmarking standards to maintain experimental integrity. Crucially, the design of the retraining feature prioritizes 'ease of use' for operators while strictly enforcing a 'human-in-the-loop' workflow during the evaluation stages to guarantee model reliability.

5.1 Retraining Process

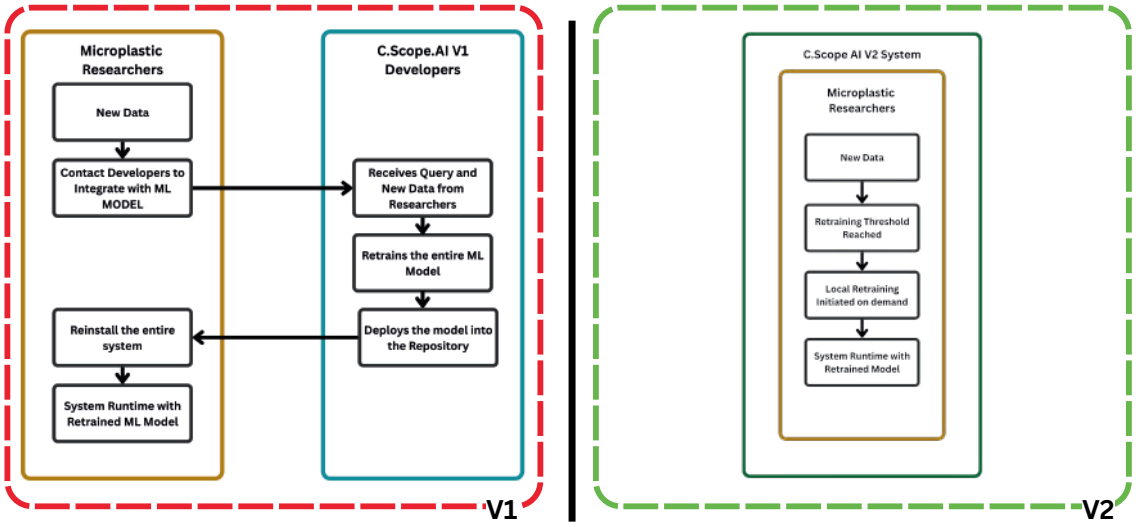


Fig. 35. Retraining Process Comparison | C.Scope.AI V1 (left) , C.Scope.AI V2 (right)

Figure 35 contrasts the retraining architectures of C.Scope.AI V1 and V2. The V1 workflow is characterized by a decoupled, bipartite dependency, where researchers must manually transmit data to developers for off-site retraining and subsequently reinstall the entire system to apply updates. This dependency introduces significant latency and operational friction. In contrast, V2 encapsulates the entire pipeline within a local, self-contained system. By automating the transition from data acquisition to on-demand local retraining, V2 eliminates external developer reliance, thereby resolving a critical bottleneck that previously hindered the system's scalability and widespread adoption.

5.1.1 Retraining Hub

For the implementation of the V2 retraining process, usability was established as the primary design objective. To achieve this, the strategy focused on centralization: the entire retraining workflow was consolidated into a unified dashboard within the C.Scope.AI application. This 'single-window' approach eliminates the need for context switching, ensuring a seamless experience for the operator.

Fig. 36. Hierarchy of the Retraining Feature *(Left)*, Retraining Interface *(Right)*

The left panel of Fig. 36 illustrates how the retraining workflow is encapsulated within a single window to ensure usability. This interface serves as a centralized hub: it houses the front-end controls for user interaction while seamlessly orchestrating the underlying back-end retraining pipeline.

As shown in the right panel of Fig. 36, the Retraining UI employs a minimalist design to prioritize ease of use and streamline productivity-focused functions. The interface was implemented using PyQt5. This framework was selected not only to maintain architectural consistency with the main application but also for its robust platform compatibility and professional widget set. Consequently, the UI utilizes PyQt5's native libraries to render key interaction elements, such as the model selection drop-down and execution buttons.

5.1.2 Retraining Pipeline

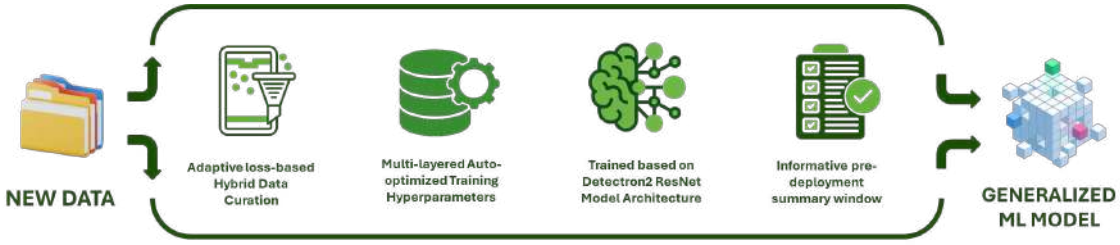


Fig. 37. Simplified Retraining Pipeline Diagram

The retraining pipeline functions as the back-end module encompassing the core logic of the retraining process. As depicted in Fig. 37, this pipeline operates as a linear, multi-stage workflow. C.Scope.AI's pipeline comprises six critical stages: (1) Environment & Model Setup, (2) Dataset Mining, (3) Data Filtering, (4) Data Combination, (5) Training Data Preparation, and (6) Training & Evaluation. Each stage is fully automated to ensure the system remains user-friendly and minimizes manual intervention.

Environment & Model Setup

This initial stage establishes the operational environment and locates the designated 'champion' model. The process begins by parsing JSON-formatted user settings to identify the specific model selected for retraining. To prevent data leakage or cross-training, Binary and Multiclass models are stored in distinct directories; the system employs the glob library to locate the correct model files via pattern matching. Additionally, a retraining timestamp is generated using Python's datetime module for version control and logging. The stage concludes by parsing the model's specific hyperparameters from YAML configuration files.

Dataset Mining

This stage evaluates the current state of the target dataset and, if applicable, the anchor dataset. The system spawns a dedicated subprocess utilizing PyTorch and Detectron2 to perform inference and calculate the loss metric for each individual image. Given the computationally intensive nature of this operation, the system automatically detects and leverages NVIDIA GPU acceleration when available. The resulting loss data is serialized into a JSON format, which serves as a critical input for the subsequent retraining steps.

Data Filtering

This stage implements a filtering mechanism designed to isolate high-value training samples based on difficulty classification. To determine which images provide a meaningful learning signal, the system employs a Hybrid Filtering Strategy. This approach synthesizes statistical loss metrics from both the anchor and new datasets to compute a dynamic selection threshold. Implemented using NumPy for efficient numerical computation, the algorithm retains only those images where the inference loss exceeds this threshold, effectively prioritizing 'hard' examples. The stage concludes with a PyQt5-based validation interface, presenting the user with mean loss comparisons to ensure informed approval before the pipeline proceeds.

Data Combination

This stage integrates the filtered new data into the anchor dataset. When 'auto-combine' is active, the system uses Python's shutil module to safely copy images from the queue to the main training directory, using OpenCV to validate the images during transfer. For annotations, the system parses the COCO-formatted JSON files from both datasets. It merges them by remapping IDs, incrementing the new IDs based on the existing maximums, to prevent database conflicts. The final merged data is saved as a formatted JSON file. Finally, the system uses SQLite3 to remove processed entries from the database and deletes old cache files, ensuring that difficulty scores are recalculated for the newly expanded dataset.

Training Data Preparation

This stage constructs the final training dataset by merging anchor data with new data based on specific ratios and difficulty distributions. First, the system uses the data curation subprocess to generate a manifest that maps each anchor image to a difficulty category (hard, medium, or easy). These categories are defined by adaptive thresholds calculated using NumPy. To ensure the dataset is reproducible, the system uses a deterministic sampling algorithm, seeded by Python's random module, to select anchor images that match the desired difficulty mix. Next, it resolves all image paths using pathlib for cross-platform compatibility and merges the annotations into a single file that complies with the standard COCO object detection format. Finally, the system calculates the optimal training duration (iterations) using math.ceil and adjusts the learning rate milestones proportionally to ensure the model converges correctly regardless of the dataset size.

Training & Evaluation

This final stage orchestrates the model training and performance assessment. The system initiates train.py as an encapsulated subprocess, injecting dynamically computed hyperparameters such as iteration count, learning rate schedules, and warmup periods. The training logic utilizes Detectron2 (built on PyTorch) and employs a CustomTrainer class to implement specialized optimization strategies for stable convergence. To ensure consistency, the system enforces a full deterministic mode while leveraging CUDA acceleration for performance. During execution, the system parses standard output (stdout) using regular expressions to extract real-time loss metrics, which are relayed to the UI via PyQt5 signals. Upon completion, the benchmark.py module evaluates both the champion and challenger models using the COCOEvaluator. The resulting Average Precision (AP) metrics are presented in a ComparisonDialog, facilitating an informed deployment decision.

5.1.3 Pre-Retraining

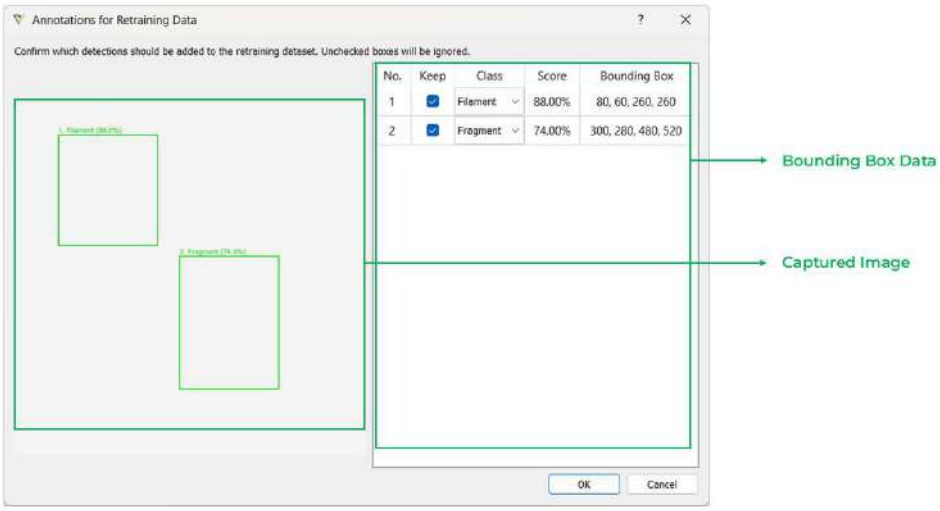


Fig. 38. Annotation Window

High-fidelity data annotation is foundational to the retraining process, providing the 'ground truth' required for the model to refine its pattern recognition and prediction capabilities. While the Version 1 prototype relied on Roboflow, an external AI-assisted annotation tool, Version 2 prioritizes a standalone architecture to eliminate external dependencies. Consequently, a custom Annotation Window presented in Fig. 38 was developed to facilitate onsite annotation. Captured images with annotation are archived in `retrain_images.db`, a database schema established during the first iteration. This database is utilized for retraining the models. To mitigate inference errors, the interface employs a human-in-the-loop methodology: users can validate True Positives, discard False Positives, and correct misclassifications. This manual intervention ensures that model deficiencies are addressed directly by expert verification.

5.1.4 Post-Retraining

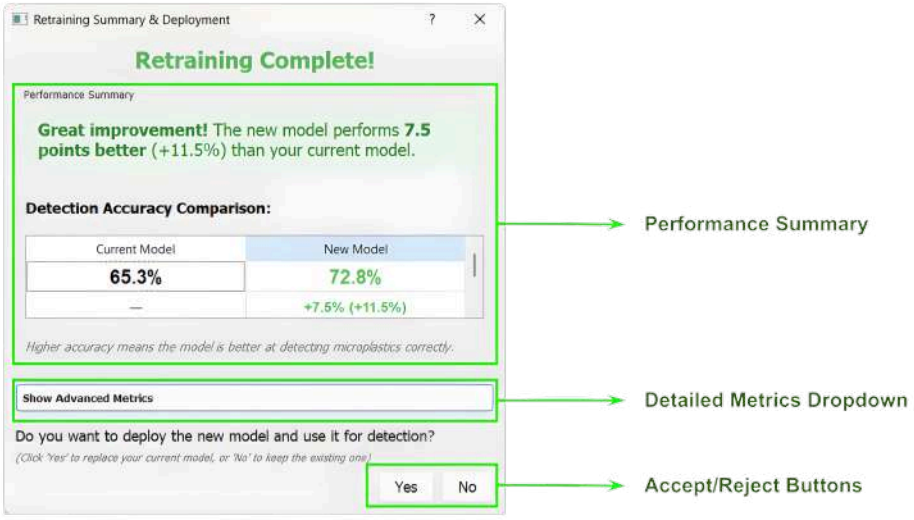


Fig. 39. Retraining Summary & Deployment Window

Upon the conclusion of the retraining pipeline, the system presents the summary interface depicted in Fig. 39. This component represents the critical 'human-in-the-loop' validation stage, granting the user final authority over model deployment. The window facilitates a comprehensive comparative analysis by ingesting evaluation dictionaries from both the champion and challenger models and rendering the COCO metrics in an organized visual hierarchy. If the user approves the update, the deployment logic utilizes Python's shutil module to promote the challenger model by copying its directory and configuration to the production environment. This validation and deployment workflow is standardized, functioning identically for both Multiclass and Binary architectures.

5.1.5 Retraining Hyperparameters & Techniques

While the preceding section established the structural framework of the C.Scope.AI V2 pipeline, the system's ultimate detection capability is governed by the granular configuration of its learning parameters. A robust pipeline structure alone cannot guarantee convergence or generalization; it requires precise optimization to function effectively. Consequently, this section details the specific hyperparameters and architectural techniques, ranging from learning rate schedules to progressive unfreezing, that were rigorously tuned to maximize performance. These interventions are critical for transforming the raw retraining workflow into a stable, high-precision instrument capable of adapting to new microplastic datasets.

Hyperparameter/Technique	Value
Base Learning Rate	Hyperparameter Optimized
Max Iterations	Hyperparameter Optimized
Steps	Hyperparameter Optimized
Gamma	0.1
Weight Decay	Hyperparameter Optimized
Warmup Iterations	Hyperparameter Optimized
Warmup Factors	Hyperparameter Optimized
IMS PER BATCH	Hyperparameter Optimized
Batch Size Per Image	Hyperparameter Optimized
Progressive Unfreezing	stem = frozen res2 = frozen res 3 - 5 = base lr x 0.1
Adaptive Threshold	Percentile: Hard = Loss>80% Med = 45% >Loss >80% Easy = Loss <45%
Selection Mix	Loss-Based Benchmarking
Dynamic Iteration Scaling	target_epochs × total_images / images_per_batch
Deterministic Training	Hyperparameter Optimized
Filter Threshold	Loss-Based Benchmarking
Anchor Ratio	Loss-Based Benchmarking

Table VII. Retraining Hyperparameter & Technique Table

Training Hyperparameters

The training configuration uses a specific set of hyperparameters derived from optimization experiments. The IMS PER BATCH is set to 3, meaning the model processes three images at a time; this small batch size balances memory usage with accurate model updates. The system uses an ultra-low Base Learning Rate to fine-tune the pre-trained weights. This slow approach allows the model to learn new microplastic features without 'forgetting' or overwriting the useful patterns it learned previously.

The training schedule is defined by the Max Iteration. To improve accuracy as the model finishes training, the STEPS parameter reduces the learning rate at roughly 67% and 95% of

the total duration. At each of these milestones, the GAMMA value (0.1) reduces the learning rate to 10% of its current speed, allowing the model to settle into an optimal state.

To prevent the model from memorizing the small dataset (overfitting), the system applies weight decay (L2 regularization), which penalizes overly complex model weights. Additionally, a Warmup Schedule is used for the first 14% of training. This gradually increases the learning rate from near-zero to the base value, preventing instability or errors that can occur if training starts too aggressively.

The BATCH SIZE PER IMAGE (set to 192) controls how many regions of interest are analyzed per image. This number ensures the model sees enough positive and negative examples to learn effectively without slowing down computation.

Progressive Unfreezing

To optimize the retraining process, the CustomTrainer employs a fine-tuning strategy that treats the neural network as a hierarchy of features. The bottom layers of the backbone (stem and res2) remain frozen to preserve fundamental visual descriptors like edges and textures. Moving up the network, the middle layers (res3-res5) are assigned a 'chilled' learning rate (10% of the base). This allows the model to gently adapt its mid-level representations without overwriting the valuable pre-trained weights, a safeguard against overfitting. Conversely, the top-level components, specifically the FPN, RPN, and ROI Heads, are trained at the full learning rate, ensuring the model rapidly adapts to the specific classification and localization requirements of the new data.

Difficulty-Based Data Selection

The system implements difficulty-based data selection, classifying images into hard, medium, and easy buckets based on model loss values from hard example mining. Adaptive thresholds are computed using percentiles of the actual loss distribution: images with loss $\geq$ 80th percentile are classified as hard, $\geq$ 45th percentile as medium, and the remainder as easy. During anchor selection, the system samples from these buckets according to the selection mix, ensuring the retraining dataset balances challenging examples that drive improvement with stable examples that reinforce existing knowledge. This data-driven approach adapts to model performance rather than relying on fixed thresholds.

Dynamic Iteration Scaling

Dynamic iteration scaling ensures consistent training coverage by automatically adjusting the number of iterations based on dataset size, maintaining approximately 32 epochs

regardless of data volume. The formula

$$iterations = \frac{(target\ epoch) \times (total\ images)}{(images\ per\ batch)}$$

derives from HPO-optimized baselines where 1358 iterations proved optimal for 126 images. The system applies bounds (minimum 500, maximum 3000) to prevent under-training on small datasets or excessive computation on large ones. This epoch-based approach eliminates the need for manual iteration tuning, automatically adapting training duration to accommodate varying retraining dataset sizes while preserving the validated learning rate decay timing.

Deterministic Training

The system enforces a Deterministic Training protocol using a fixed global seed (default: 1337). This configuration initializes all random number generators to a synchronized state via torch.manual seed() for CPU operations, torch.cuda.manual seed all() for GPU acceleration, and Python's native random.seed(). Consequently, stochastic processes such as data shuffling yield identical outputs across executions, facilitating reliable model comparison and pipeline debugging.

Filter Threshold

Retraining employs a K Threshold Filter to discard low-value training data. By analyzing the loss statistics of the anchor dataset, the system computes a dynamic threshold:

$$T = Mean - (K \times StdDev)$$

New images falling below this threshold are classified as 'easy' and removed to prevent overfitting to well-learned patterns. The value will be determined by the loss-based benchmarking.

Anchor Ratio

The system utilizes an Anchor Ratio to mitigate catastrophic forgetting by blending original dataset samples with new data. Defined as a percentage of the new data quantity, a 150% ratio implies that for every 100 new images, 150 anchor images are selected via difficulty stratification. Loss-based benchmarking results will dictate the optimal value for knowledge retention. The optimal value will be hard-coded into the system.

5.1.6 Retraining Model Curation

With the optimal hyperparameters and transfer learning techniques established, the validation phase encounters a critical operational constraint: the unavailability of native Version 2 field data. Consequently, the initial retraining experiments are restricted to the legacy datasets utilized by the Version 1 system. This reliance on V1 data necessitates a rigorous Model Curation strategy. This section details how a specific test model was selected and curated from the V1 repository to serve as a reliable baseline, ensuring that the V2 retraining pipeline is validated against established ground truth before deployment in the field.

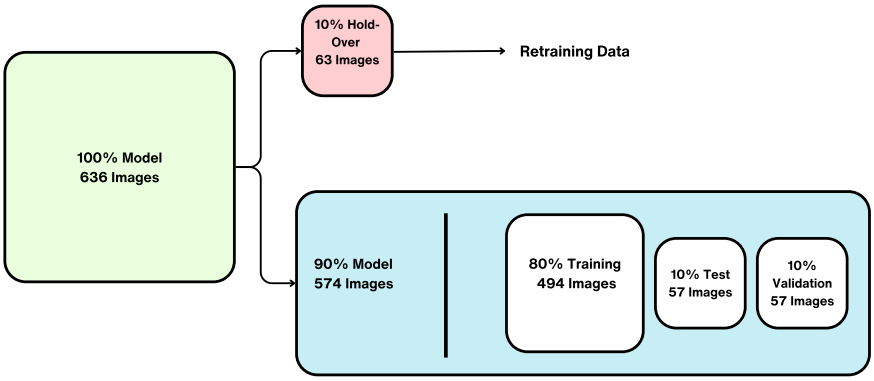


Fig. 40. Model Curation Diagram

As shown in Fig. 40, a data partitioning strategy was adopted to simulate the retraining loop using the existing dataset of 636 images. To replicate a real-world scenario, the full dataset was split 90/10. The 10% subset served as the proxy for 'new' data, while the remaining 90% formed the anchor dataset. This anchor dataset was subsequently organized into an 80/10/10 distribution for training, testing, and validation. This approach allowed for the validation of the retraining pipeline without external data ingestion and was applied to both model architectures.

5.1.7 Hyperparameter Optimization

To mitigate the labor-intensive nature of manual tuning, the system implements an Automated Hyperparameter Optimization (HPO) process. The Optuna framework is utilized to systematically search for optimal training configurations. The optimizer explores a defined search space encompassing the learning rate, maximum iterations, weight decay, scheduler

types, and warmup parameters. To ensure experimental reproducibility, each trial is executed with a fixed random seed; this value is an arbitrary constant chosen to guarantee deterministic results across runs. The performance of each configuration is quantified using Average Precision (AP) benchmarking. Optuna's Tree-structured Parzen Estimator (TPE) sampler drives the optimization process, utilizing results from completed trials to intelligently narrow the search toward the most promising hyperparameter combinations.

5.2 Retraining Benchmarking

A robust benchmarking framework is essential to validate the efficacy of the local retraining pipeline. In a system designed for field adaptation, it is insufficient to merely complete a training cycle; the resulting model must be rigorously evaluated to ensure it meets performance standards without compromising pre-existing capabilities. Consequently, this section outlines the dual-metric evaluation strategy employed to qualify retrained models. The analysis begins with Average Precision Based Benchmarking, which quantifies the functional detection accuracy and recall capabilities of the system. This is complemented by Loss-Based Benchmarking, a diagnostic approach used to assess the stability, confidence, and convergence quality of the optimization process.

5.2.1 Average Precision Based Benchmarking

The Average Precision (AP) benchmarking system uses Detectron2's COCOEvaluator to quantify model performance against COCO-format ground truth annotations. The AP benchmark script registers a test dataset, runs inference using inference_on_dataset(), and computes COCO metrics that measure detection quality across multiple Intersection over Union (IoU) thresholds.

Core Metrics

Intersection over Union (IoU)

Measures overlap between predicted box B_p and ground truth box B_{gt}:

$$IoU = \frac{|B_p \cap B_{gt}|}{|B_p \cup B_{gt}|}$$

A prediction is a True Positive if IoU $\geq$ threshold otherwise False Positive.

Precision and Recall

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

Where: TP = True Positives (correct detections) FP = False Positives (incorrect detections) FN = False Negatives (missed ground truths)

- AP (Average Precision): The primary metric, computed by averaging precision across IoU thresholds from 0.50 to 0.95 in 0.05 increments, then averaging across all classes. Represents overall detection quality.
- AP50: Precision at IoU $\geq$ 0.50, a lenient threshold indicating whether detections roughly overlap ground truth.
- AP75: Precision at IoU $\geq$ 0.75, a stricter threshold requiring precise localization.
- APs/APm/APl: AP for small, medium, and large objects, revealing scale-specific performance.

Evaluation Process

For each image, the model produces bounding box predictions with confidence scores. These are matched against ground truth boxes using IoU (Intersection over Union). A precision-recall curve is computed by varying the confidence threshold, and AP is the area under this curve. The system computes per-class AP then averages to obtain mAP (mean Average Precision).

Champion vs Challenger Comparison

$$Improvement = AP_{challenger} - AP_{champion}$$

$$Improvement\% = \frac{AP_{challenger} - AP_{champion}}{AP_{champion}} \times 100$$

The ComparisonDialog ingests evaluation dictionaries from both models, displaying AP, AP50, and AP75 with color-coded improvement indicators (green for gains, red for declines). It estimates practical impact by converting AP50 scores to approximate "images detected well" counts, making metrics accessible to non-technical users while preserving detailed breakdowns in the collapsible advanced section.

5.2.2 Loss-Based Benchmarking

Loss-based benchmarking quantifies image difficulty by computing the training loss for each individual sample. In contrast to AP-based benchmarking, which evaluates overall detection accuracy, this approach measures the divergence between the model's prediction and the ground truth. Consequently, a higher loss indicates a greater error magnitude, signifying that the specific sample is more challenging for the model to process.

Process Overview

The helper script runs the model in training mode to access loss computation. It starts with loading the dataset from the COCO formatted annotations. Then it performs forward pass where each image is processed. Then loss extraction is performed where classification and localization loss values are collected. Loss Aggregation is then performed combining loss components into a total loss per image. All images are then ranked in a descending manner. Lastly, an automated light data analysis is performed where distribution metrics such as mean, standard deviation, and percentiles are processed.

Loss Components

Component	Description
loss_cls	Region Proposal Network (RPN) classification loss
loss_box_reg	RPN bounding box regression loss
loss_cls (ROI)	Region of Interest (ROI) classification loss
loss_box_reg (ROI)	ROI head box refinement loss

Table VIII. Table of Loss Components

Total Loss (Per Image)

$$L_{total} = L_{rpn\_cls} + L_{rpn\_box} + L_{roi\_cls} + L_{roi\_box}$$

Loss Metrics

Mean Loss Reduction (MLR)

$$MLR = \mu_{before} - \mu_{after}$$

Where:

$$\mu_{\text{before}} = \textit{mean loss before training}$$
$$\mu_{\text{after}} = \textit{mean loss after training}$$

Normalized MLR (for composite score):

$$MLR_{norm} = \frac{\mu_{before} - \mu_{after}}{\mu_{before}} = \frac{MLR}{\mu_{before}}$$

Mean Loss Reduction (MLR) is defined as the absolute decrease in average loss following the training process. This metric quantifies the degree of optimization, effectively measuring how much the model's 'difficulty' in processing the dataset has diminished. A positive MLR confirms model convergence, where a larger magnitude signifies a more substantial improvement in prediction accuracy and a corresponding reduction in error rates.

Hard Example Reduction Rate (HERR)

$$HERR = \frac{N_{hard}^{before} - N_{hard}^{after}}{N_{hard}^{before}}$$

Where:

N_{hard}^{before} = *count of hard examples before* training
N_{hard}^{after} = *count of hard examples after training*

Hard Example Reduction Rate (HERR) quantifies the decline in the volume of high-difficulty samples following the training process. Specifically, it tracks the proportion of images that initially exceeded the difficulty threshold but were successfully optimized to fall below it. For instance, a HERR of 0.5 indicates that 50% of the previously classified 'hard' examples have been resolved, while a value of 1.0 signifies complete resolution. This metric is critical as it validates that the model is optimizing its weakest areas rather than merely overfitting to easy samples.

Stability

$$Stability = 1 - \frac{\sigma_{after}}{\sigma_{before}}$$

Where:

σ_{before} = standard deviation of losses before training
σ_{after} = standard deviation of losses after training

Stability quantifies the reduction in loss variance following the training process. It serves as a measure of optimization consistency, distinguishing between uniform learning and concentrated performance spikes. A higher stability score indicates that the distribution of loss values has become more homogeneous (narrower spread), implying that the model has improved evenly across the dataset. Crucially, this reduction in variance is a strong indicator of generalized learning, whereas inconsistent improvements often signal overfitting to specific subsets.

Composite Score

$$Composite = (0.45 \times MLR_{norm}) + (0.35 \times HERR) + (0.20 \times Stability)$$

The Composite Score aggregates MLR, HERR, and Stability into a single ranking using the following weights:

Component	Weight	Rationale
MLR	45%	Overall all lost reduction
HERR	35%	Improvement of hard examples
Stability	20%	Quality assurance and consistency

Table IX. Table of Composite Scoring Components

Mean Loss Reduction (MLR) is allocated the highest weight (45%) as it serves as the primary indicator of global optimization. A model that achieves a significant reduction in mean loss across the entire dataset demonstrates a fundamental enhancement in its overall detection capability.

Hard Example Reduction Rate (HERR) receives a substantial weight (35%) because addressing difficult edge cases is central to the objectives of loss-reduction learning. Improvements limited to easy examples offer minimal practical utility; real-world robustness depends on resolving the failure cases associated with hard samples.

Stability functions as a diagnostic metric rather than a primary performance driver, receiving a weight of 20%. While consistency is desirable, it is subordinate to error reduction. A configuration yielding high MLR and HERR values represents a genuine performance breakthrough, even if the stability of the convergence is moderate.

5.3 Retraining Results

This section documents the retraining experiments conducted to identify the optimal hyperparameters for system deployment. The study focuses on the multiclass model, as it presents a greater optimization challenge than the binary equivalent. By solving for the higher complexity of the multiclass problem, the resulting solutions possess greater generality. Furthermore, since both models share an identical architecture, findings from these experiments are directly transferable to the binary model.

5.3.1 Baseline Establishment

The following section presents the baseline performance statistics for the multiclass model. The result tables focus on Average Precision (AP) and class-specific AP values for Filament, Film, and Fragment. Although AP_{50} and AP_{75} were computed by the evaluator, they are omitted from the comparative result tables because they followed the same trend as AP and did not provide a distinct interpretation. Complementing these metrics, the loss-based benchmarking includes both numerical statistics and visual distributions to illustrate sample difficulty.

Category	Value
AP	79.68
AP-Filament	79.96
AP-Film	75.96
AP-Fragment	83.11

Table X. AP-Based Benchmark Baseline Score - Anchor Dataset

Table X displays the baseline scores for the dataset on which the base model was trained. The high values confirm that the initial training script (Version 1) achieved successful convergence. For future reference, any decrease in these scores during retraining will be analyzed to determine if it stems from better model generalization or strictly from forgetting previously learned features.

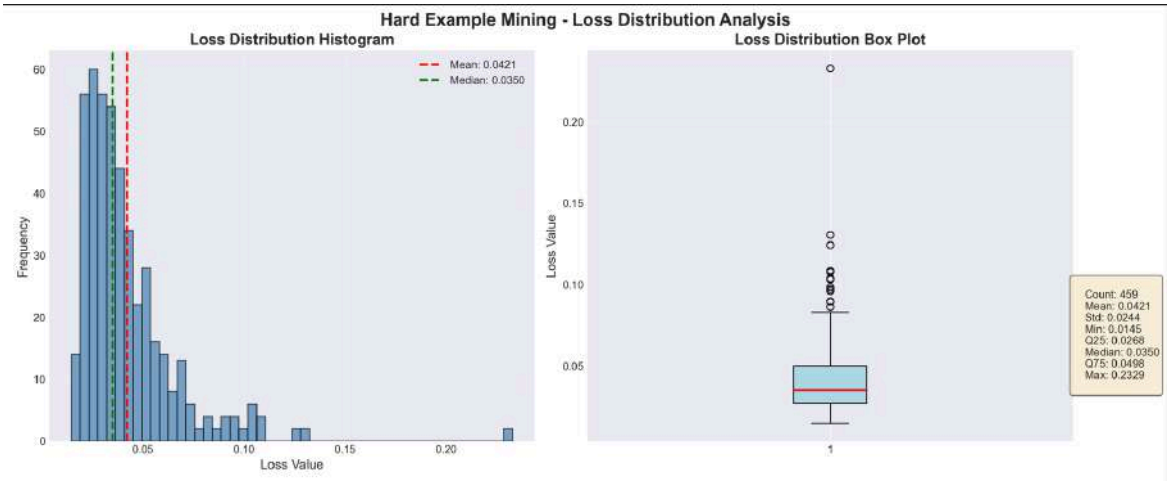


Fig. 41. Baseline Image Distribution Anchor Histogram (Left) & Box Plot (Right)

As illustrated in Fig. 41, the loss values for the anchor dataset are predominantly low, with the vast majority falling below a threshold of 0.15. This distribution serves as a critical baseline; subsequent deviations observed during referencing will be instrumental in distinguishing between model generalization and catastrophic forgetting.

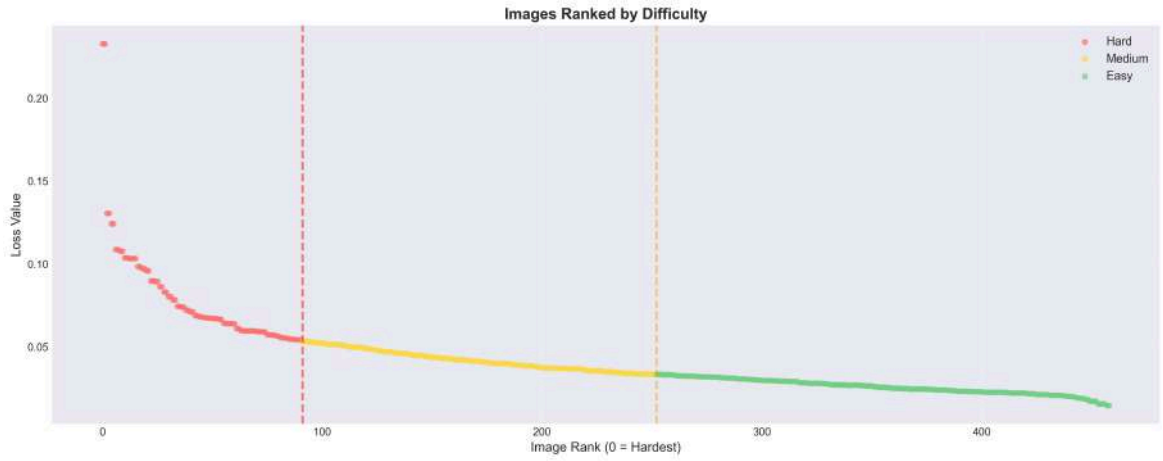


Fig. 42. Baseline Image Ranked Distribution Anchor

As depicted in Fig. 42, the image difficulty rank distribution follows a near-linear progression across the low-to-moderate difficulty range, whereas the upper tail exhibits an exponential increase. Theoretically, an ideal model would yield a horizontal distribution, indicating uniform performance across all samples regardless of complexity. Consequently, the optimization objective is to flatten this curve, minimizing the disparity between easy and hard samples, while acknowledging that statistical outliers are inevitable.

Category	Value
AP	41.19
AP-Filament	31.11
AP-Film	48.60
AP-Fragment	43.85

Table XI. AP-Based Benchmark Baseline Score - Retrain Dataset

The AP benchmark scores for the retrain dataset indicate a performance disparity favoring the 'Film' class, which surpasses all other microplastic categories in Average Precision. This suggests a potential class imbalance or a 'Film-heavy' distribution within the training data. Overall, while the model demonstrates a baseline level of generalization, the variance in class-specific performance indicates significant scope for further optimization.

Fig. 43. Baseline Image Distribution Retrain Histogram (Left) & Box Plot (Right)

As illustrated in Fig. 43, the retraining dataset exhibits a semi-linear difficulty distribution. The calculated mean loss is 0.11, which is significantly higher than the ideal baseline of 0.04

observed in the anchor data. This discrepancy highlights a substantial margin for optimization during the retraining phase.

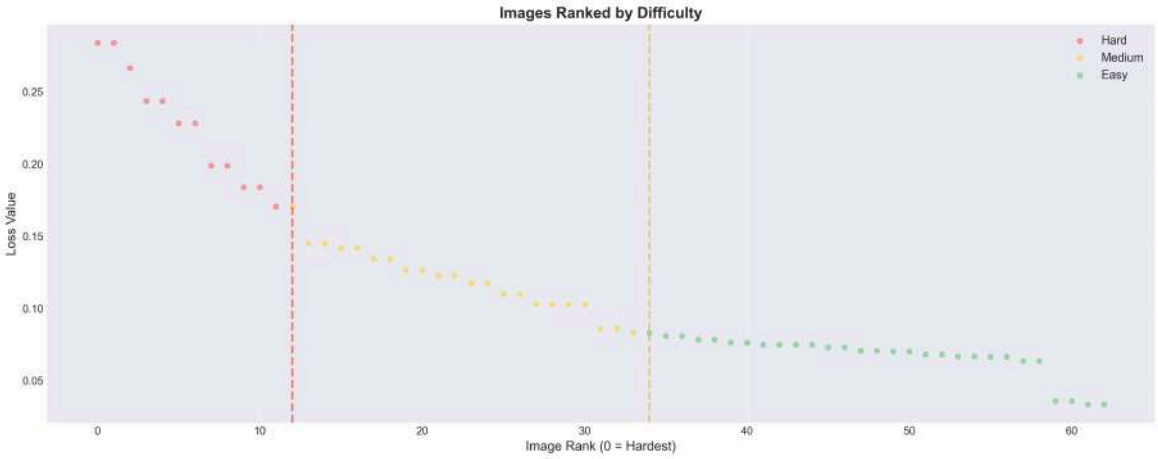


Fig. 44. Baseline Image Ranked Distribution Retrain

Figure 44 corroborates the quasi-linear nature of the difficulty distribution. The curve remains near-horizontal across the 'easy' sample range (indicating a plateau of low loss), transitions into a gradual inflection at the medium range, and culminates in a steep, exponential ascent for the 'hard' samples.

Category	Value
AP	44.23
AP-Filament	45.46
AP-Film	40.40
AP-Fragment	46.82

Table XII. AP-Based Benchmark Baseline Score - Test Dataset

The test dataset functions as the most balanced of the two validation sets, exhibiting comparable performance scores across all three classifications (45.46, 40.40, and 46.82). Consequently, significant performance fluctuations on this dataset will serve as a robust indicator of either model generalization or catastrophic forgetting.

Fig. 45. Baseline Image Distribution Test Histogram (Left) & Box Plot (Right)

Similar to the retraining dataset, the distribution histogram in Fig. 45 exhibits a comparable structure, reinforcing the evidence for the model's generalization capabilities. While the test dataset appears marginally less challenging, yielding a mean loss of 0.09 compared to the retraining set's 0.11, it remains significantly distinct from the ideal baseline of 0.04 established by the anchor dataset.

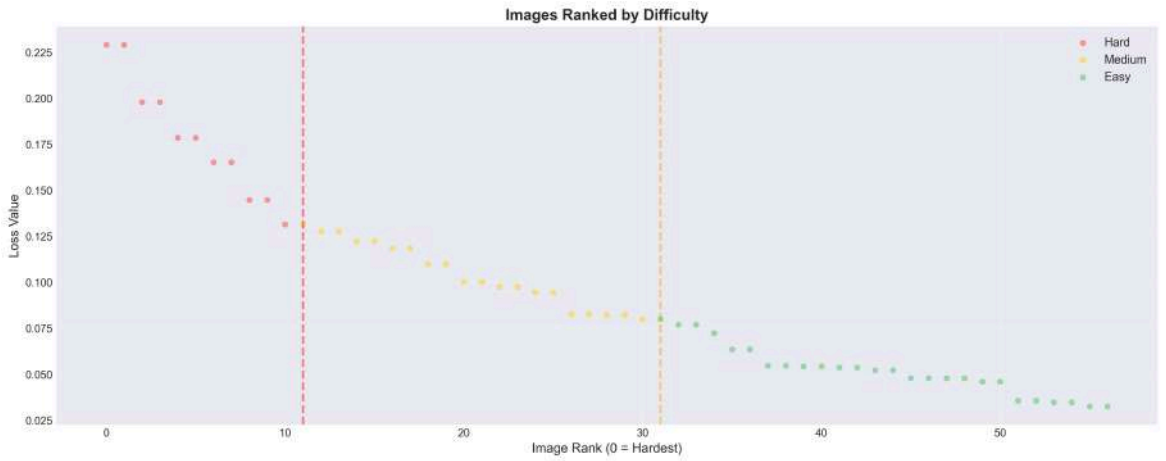


Fig. 46. Baseline Image Ranked Distribution Test

Figure 46 corroborates the earlier observation regarding the quasi-linear nature of the difficulty distribution. The curve exhibits a trajectory nearly identical to that of the base model on the retraining set. This structural consistency serves to validate the model's generalization capabilities, demonstrating stable performance patterns despite the inherent variation between the datasets.

Category	Value
AP	31.75
AP-Filament	35.78
AP-Film	16.25
AP-Fragment	43.23

Table XIII. AP-Based Benchmark Baseline Score - Validation Dataset

The validation dataset presents a distinct challenge, characterized by a high concentration of difficult 'Film' samples. This stands in sharp contrast to the retraining dataset, where the 'Film' class yielded the highest detection accuracy. Given this discrepancy, and the likelihood that the model has already optimized for the easier 'Film' features, metrics for this class are expected to remain stable. Consequently, performance improvements in the 'Filament' class serve as the primary indicator of effective training and model generalization.

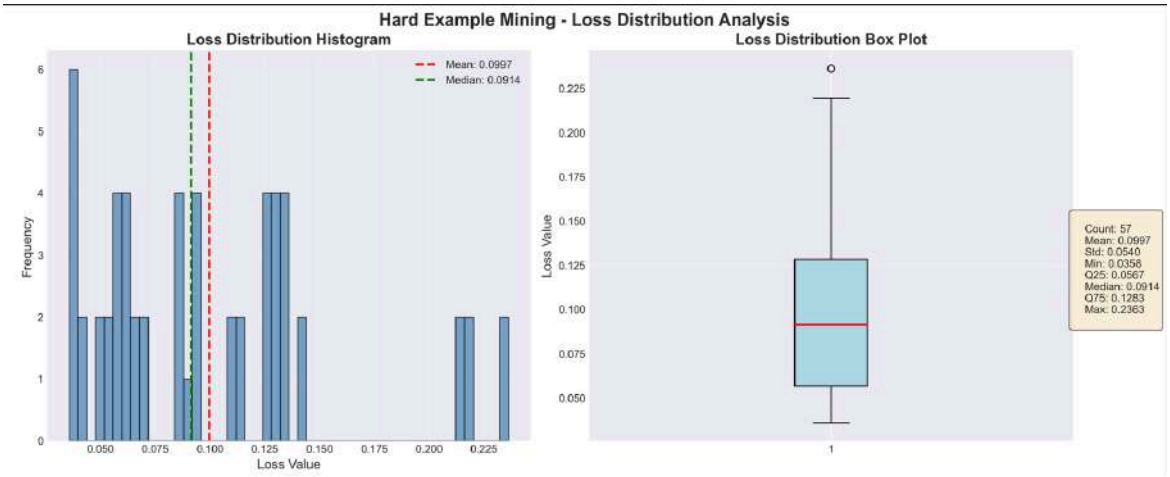


Fig. 47. Baseline Image Distribution Validation Histogram (Left) & Box Plot (Right)

Figure 47 confirms the anticipated concentration of difficult 'Film' samples. The distribution exhibits a distinct bimodal pattern: a primary cluster of low-loss samples (< 0.15),

a significantly sparse interval between 0.15 and 0.20, and a secondary peak of high-loss samples in the 0.20–0.225 range.

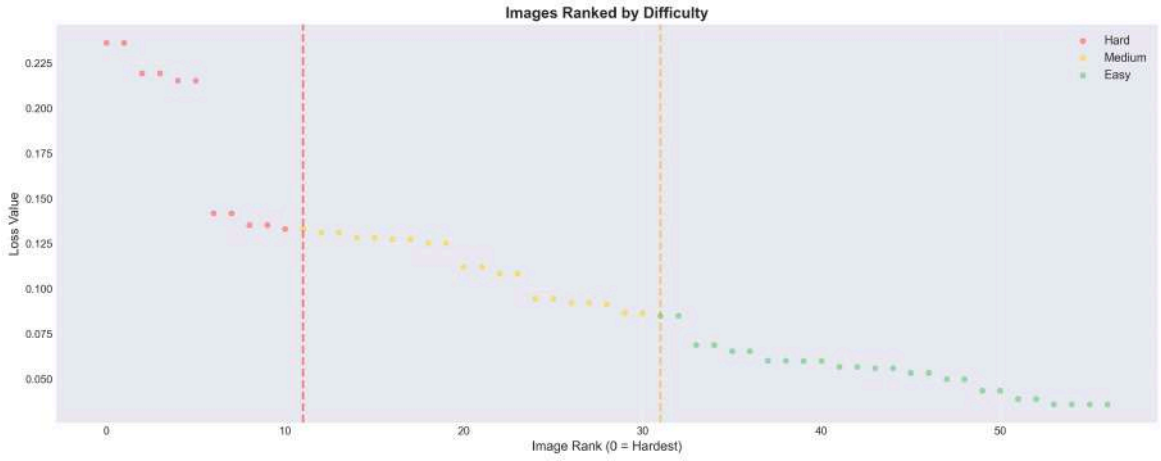


Fig. 48. Baseline Image Ranked Distribution Validation

Figure 48 further corroborates the distinct distribution observed in the histogram. The difficulty curve remains smooth and gradual across the low-to-medium range, representing the model's generalized performance. However, this stability is interrupted by a sharp discontinuity, a drastic vertical jump, transitioning into the upper-difficulty range. This sudden shift demarcates the boundary between the model's generalized capability and the specific 'hard' outliers.

Baseline Establishment Summary

The baseline analysis confirms that the base model achieved strong convergence on the Anchor dataset, establishing a low-loss reference (< 0.15) critical for monitoring catastrophic forgetting. Benchmarking reveals significant optimization potential in the Retraining dataset, which exhibits a higher mean loss of 0.11 and a performance bias toward the 'Film' class. The Test dataset functions as the primary generalization indicator due to its balanced class performance, while the Validation dataset presents a distinct bimodal challenge characterized by high-difficulty 'Film' outliers. Consequently, the forthcoming retraining phase aims to flatten the difficulty distribution curve, minimizing the disparity between easy and hard samples, and specifically enhance 'Filament' detection to ensure robust generalization across all levels of image complexity.

These baseline AP and loss distributions serve as the reference point for all subsequent optimization, transfer-learning, and loss-based comparisons.

5.3.2 Hyperparameter Optimization

The hyperparameter optimization process was executed using the Optuna framework, comprising 40 trials distributed across two random seeds (1337 and 2025). The total computational runtime was 11 hours and 10 minutes using CUDA acceleration, averaging approximately 16 minutes and 46 seconds per trial. The experiments utilized the base multiclass model, with the 'retraining' dataset serving as the optimization target and the 'test/validation' datasets acting as generalization indicators. For the evaluation metric, we prioritized AP-based benchmarking over loss-based metrics, as Average Precision provides a more definitive measure of the hyperparameters' impact on actual detection performance.

Hyperparameter Optimization History

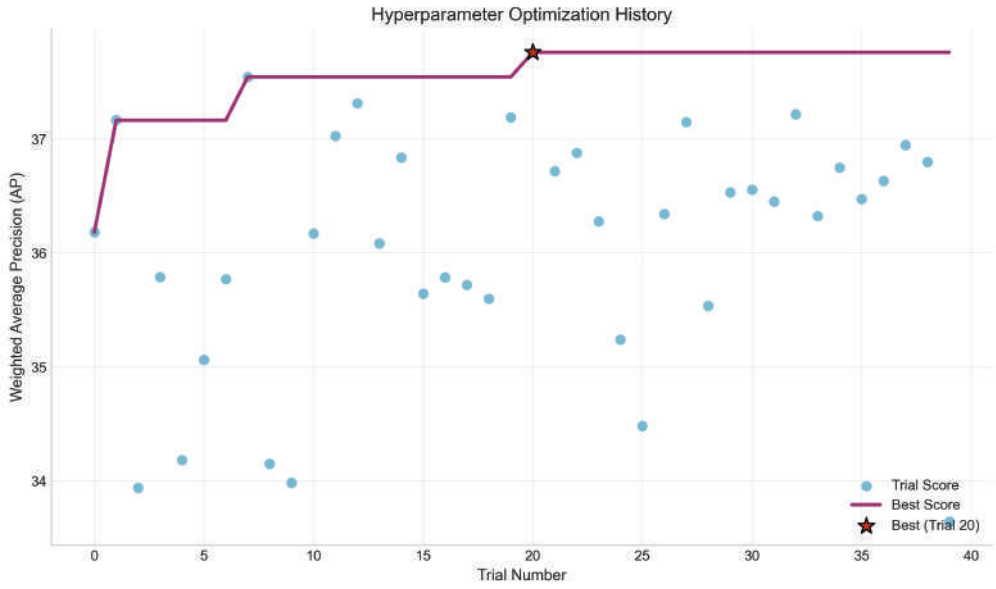


Fig. 49. Hyperparameter Optimization History Diagram

As illustrated in Fig. 49, the hyperparameter optimization trajectory achieved its maximum at the 20th trial. While the experiment was designed with a budget of 40 trials to utilize an overnight runtime window, the results indicate a saturation point at the halfway mark. The subsequent plateau observed from Trial 21 to 40 suggests that the additional

computational cost yielded diminishing returns. However, these latter trials serve to validate the robustness of the optimum found in Trial 20, confirming that further exploration within the search space was unlikely to yield significant gains without a substantial increase in computational resources.

Hyperparameter Importance

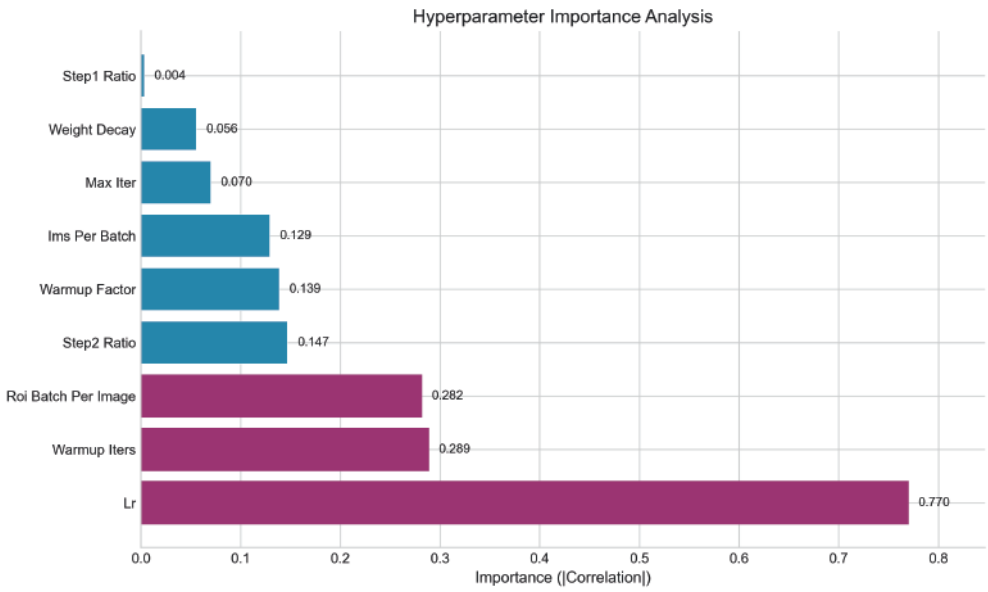


Fig. 50. Hyperparameter Importance Analysis Diagram

As illustrated in Fig. 50, the Base Learning Rate emerges as the most critical hyperparameter influencing Average Precision. Consequently, this parameter requires the highest priority during the tuning process. However, the Max Iteration parameter remains a significant variable; while secondary in impact, it governs the total computational budget and plays a crucial role in mitigating overfitting tendencies through early stopping.

Trial Performance

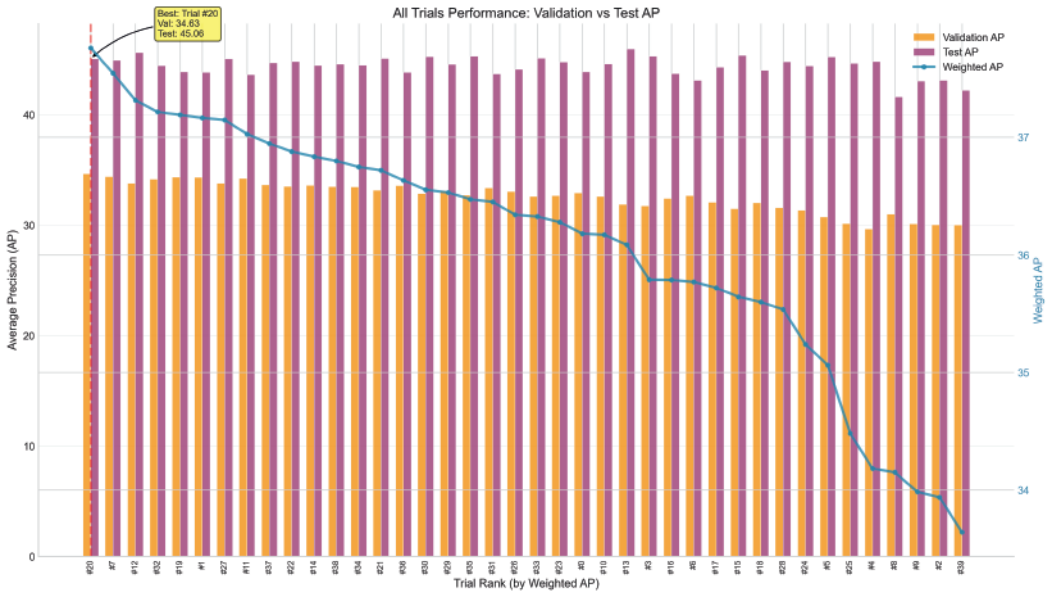


Fig. 51. All Trial Performance Ranked Diagram

As illustrated in Fig. 51, the top-performing cluster (Trials 20, 7, and 12) significantly outperforms the subsequent tier (Trials 32, 19, 1, and 27), which exhibits tightly clustered performance metrics. Furthermore, distinct performance gaps are evident between Trial 27 and Trial 11, as well as between Trial 13 and Trial 3. Consequently, these trials of interest (20, 7, 12, 27, 11, 13, and 3) have been selected for detailed analysis in the following section.

	Learning Rate	Max Iterations	Validation AP	Test AP	Weighted AP
Trial 20	0.0000151	1358	34.631	45.057	37.759
Trial 7	0.000010	1998	34.384	44.946	37.544
Trial 12	0.000015	1931	33.760	45.606	37.314
Trial 27	0.0000075	1900	33.766	45.035	37.146
Trial 11	0.0000052	2193	34.204	43.613	37.027
Trial 13	0.0000228	1844	31.868	45.921	36.084
Trial 3	0.0000781	847	31.721	45.278	35.788

Table XIV. Table for Performance of Trial of Interest

Table XIV provides a granular breakdown of the top performing trials, highlighting the advantages of Trial 20 relative to other configurations. A comparison with Trial 12 is particularly instructive: while both share nearly identical learning rates ($\approx 1.5 \times 10^{-5}$),

Trial 12's extended training duration (1931 iterations vs. 1358) resulted in a degradation of Validation AP (33.760 vs. 34.631) in exchange for only a marginal gain in Test AP. Furthermore, the disparity between Trial 27 and Trial 11 underscores the critical importance of the learning rate; despite Trial 11 utilizing more iterations, the higher learning rate of Trial 27 proved more effective across both datasets. Finally, Trials 13 and 3 demonstrate the risks of aggressive learning rates. Trial 3, in particular, suffered a significant performance penalty (Weighted AP: 35.788) due to its insufficient iteration count (847), failing to converge despite the high learning rate.

Optimal Seed

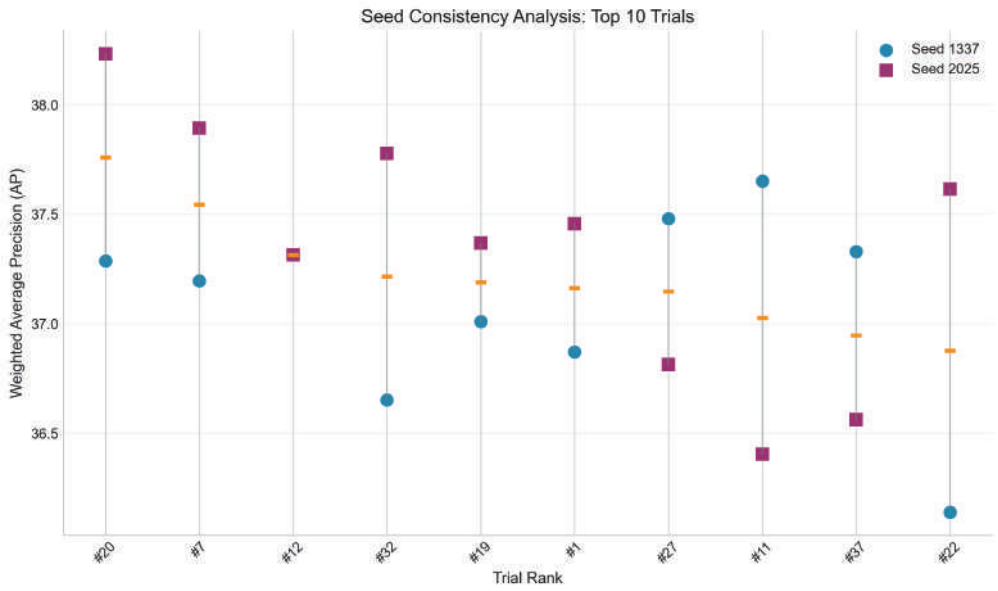


Fig. 52. Seed Consistency Analysis Top 10 Diagram

Figure 52 illustrates the impact of random seed initialization on the top 10 performing trials. The data indicates a performance bias favoring Seed 2025, which achieved superior AP scores in 6 out of 10 configurations, whereas Seed 1337 outperformed in only 3 instances. A notable case of seed sensitivity is observed in Trial 20, where the initialization resulted in a significant discrepancy of approximately 0.9 AP ($38.23 - 37.29$) between the two seeds.

Optimal Hyperparameter

Hyperparameter	Optimal Value
Base Learning Rate	0.0000151
Max Iterations	1358
Weight Decay	0.000363
Warmup Iterations	196
Warmup Factor	0.00181
Ims Per Batch	3
ROI Batch Per Image	192
Step 1 Ratio	0.6765
Step 2 Ratio	0.9496
LR Schedule Steps	[918, 1289]
Seed	2025

Table XV. Optimal Hyperparameter - Trial 20

Table XV outlines the globally optimal hyperparameter configuration derived from Trial 20. The solution is defined by a conservative Base Learning Rate of 1.51×10^{-5} and a Max Iteration count of 1358, a combination that prioritizes precise weight adjustments over rapid convergence. This granular approach is supported by a multi-step Learning Rate Schedule, where decay steps are triggered at iterations 918 and 1289 (derived from the 67.65% and 94.96% optimization ratios).Crucially, the configuration is explicitly locked to Random Seed 2025. As demonstrated in the seed consistency analysis, this specific initialization provided a statistically significant performance advantage (≈ 0.9 AP gain) over alternative seeds. By fixing the seed, this configuration ensures that the reported high-performance metrics are fully reproducible and not the result of transient stochastic variance.

Dataset	Baseline AP	Trial 20 AP	AP Gain
Test	44.23	45.057	+0.827
Validation	31.75	34.631	+2.881

Table XVI. Baseline vs Trial 20 Hyperparameter Optimization Performance

Table XVI shows that hyperparameter optimization alone improved generalization relative to the baseline. Trial 20 increased Test AP by +0.827 and Validation AP by +2.881, confirming that tuning the learning rate, iteration count, warmup, and seed improved model

performance. However, the gain remained moderate, motivating the subsequent use of transfer-learning techniques.

Hyperparameter Optimization Summary

The optimization experiments conducted via Optuna identified Trial 20 as the global optimum, achieving performance saturation at the midpoint of the 40-trial computational budget. Sensitivity analysis confirms that the Base Learning Rate serves as the primary determinant of Average Precision, with Max Iterations functioning as a secondary constraint critical for mitigating overfitting. Comparative benchmarking demonstrates that the optimal configuration (Base LR $\approx 1.51 \times 10^{-5}$, Max Iterations $= 1358$) effectively balances precise weight adjustment with training efficiency, avoiding the validation degradation observed in trials with extended durations. Furthermore, the analysis highlighted the necessity of initialization determinism; Seed 2025 demonstrated a statistically significant advantage over random initialization, contributing an approximate 0.9 AP gain. Consequently, the final deployment model strictly adheres to the Trial 20 parameter set initialized with Seed 2025 to ensure maximum reproducibility and detection stability.

5.3.3 Transfer Learning Techniques

Following the identification of optimal hyperparameter values, performance gains began to plateau, yielding results that remained insufficient for robust field deployment. Consequently, the optimization strategy shifted from parameter tuning to architectural modifications. Specifically, transfer learning techniques were implemented to leverage learned feature representations, addressing the limitations of retraining at the architectural level.

Progressive Unfreezing

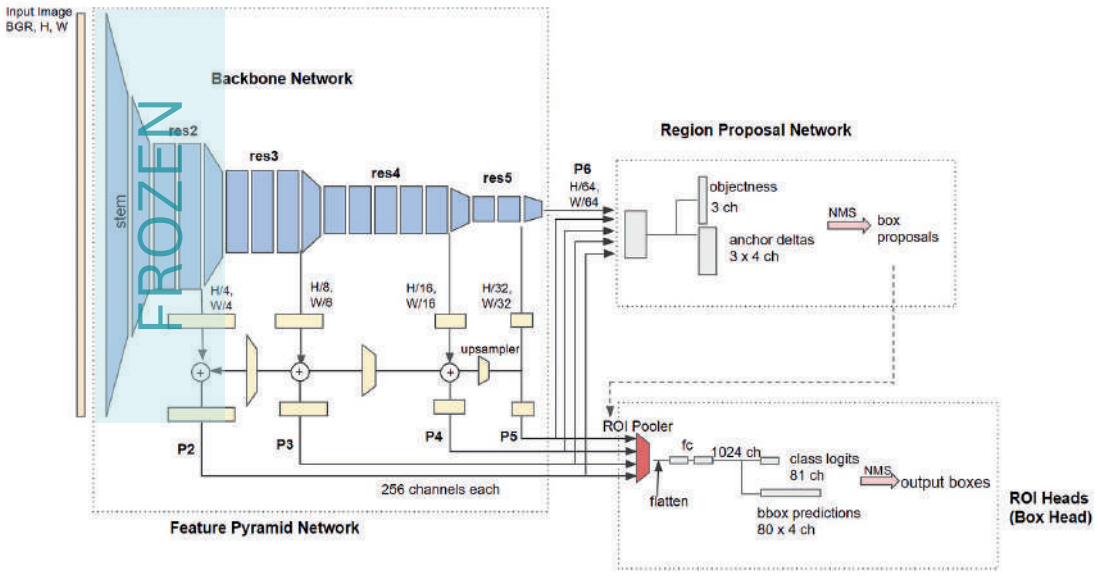


Fig. 53. Detectron2 Architecture with Frozen Backbone Diagram

The first transfer learning technique implemented was the progressive unfreezing of the Backbone Network. As illustrated in Fig. 53, the stem and res2 blocks are explicitly frozen to prevent weight updates during training. Since early layers are responsible for extracting low-level, generic features such as edges, gradients, and basic textures, these representations are universal across domains. Freezing these layers stabilizes the training process and preserves these fundamental feature extractors, thereby preventing forgetting of the model's core visual capabilities.

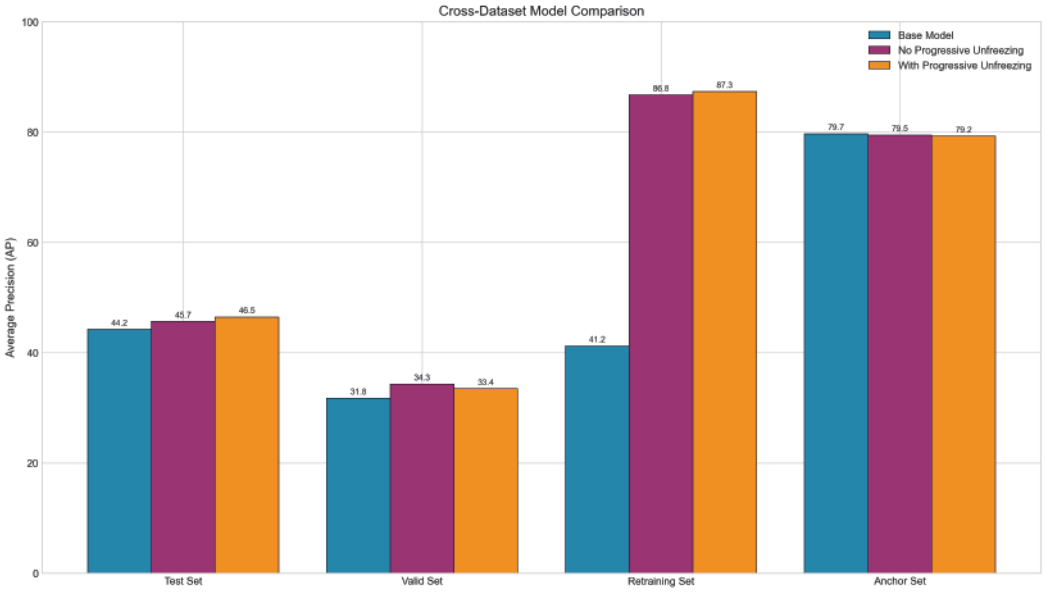


Fig. 54. Cross-Dataset Model Progressive Unfreezing Comparison Diagram

Figure 54 illustrates the efficacy of the progressive unfreezing strategy. The technique yielded a measurable improvement in generalization, boosting the Test Set AP to 46.5, surpassing both the static transfer approach (45.7) and the base model (44.2). Similarly, the Retraining Set saw peak performance with progressive unfreezing (87.3). While a slight regression was observed in the Validation Set (33.4) compared to the non-progressive approach (34.3), the method maintained high stability on the Anchor Set. The negligible drop in Anchor AP (from 79.7 to 79.2) confirms that the model successfully adapted to new data without suffering from forgetting of its original features.

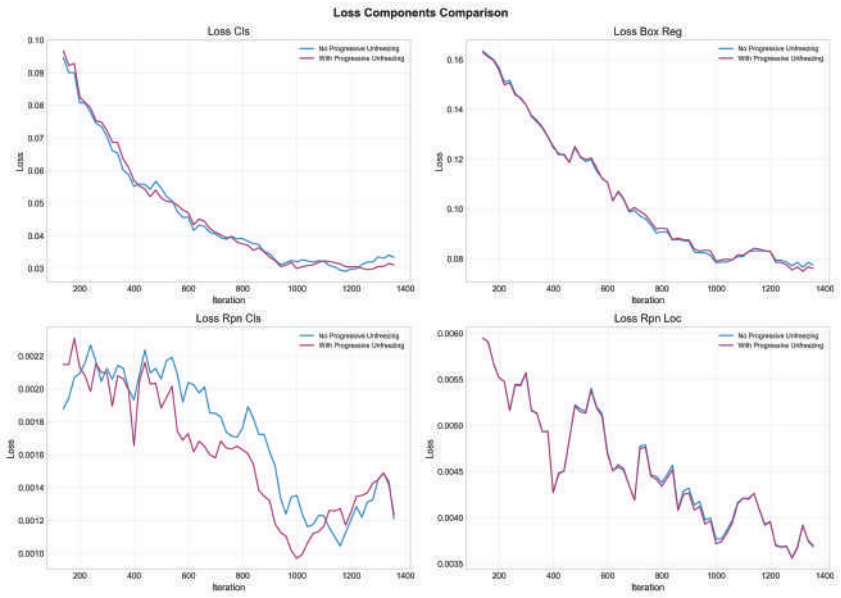


Fig. 55. Progressive Unfreezing Loss Components Loss Comparison Diagram

Figure 55 demonstrates a significantly more stable loss curve for the model trained with progressive unfreezing, particularly within the Region Proposal Network (RPN) classification task. This stability indicates that the gradual introduction of trainable parameters mitigates gradient shock, preventing the sudden destabilization of pre-learned feature representations. By slowly unlocking layers, the optimizer avoids erratic oscillations (as seen in the baseline model's RPN loss), leading to a smoother and more efficient convergence trajectory.

Differential Learning Rate

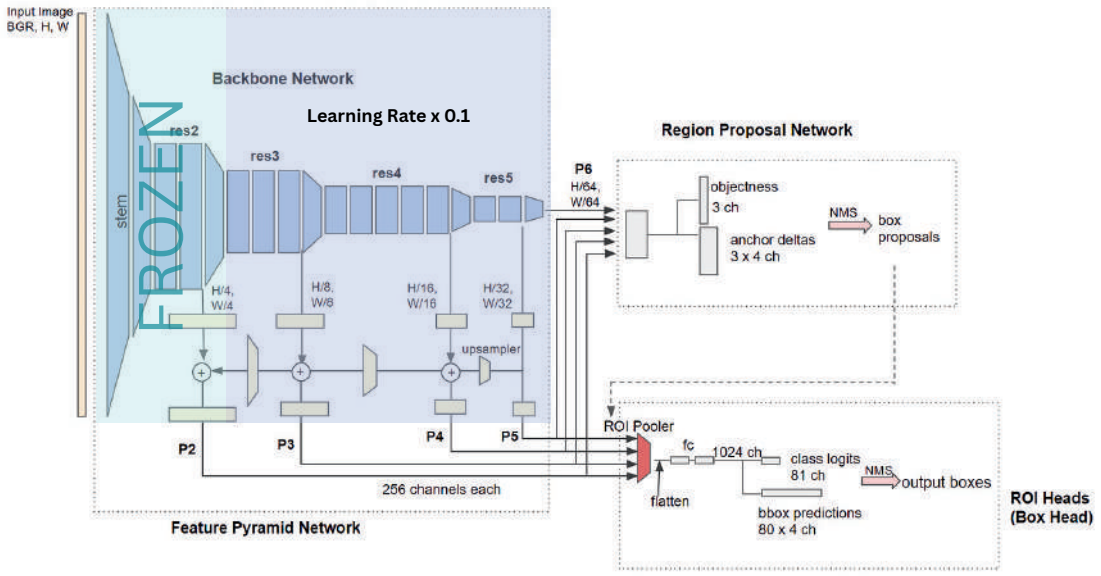


Fig. 56. Differential Learning Rate (Backbone Chilling) Diagram

The next architectural technique involved Differential Learning Rate, or 'Backbone Chilling.' By applying a 0.1x multiplier to the backbone's learning rate, this technique restricts the magnitude of weight updates in the feature extractor. This strategy is advantageous as it facilitates gentle domain adaptation, allowing the backbone to fine-tune for microplastic detection while strictly limiting the risk of forgetting essential trained visual patterns.

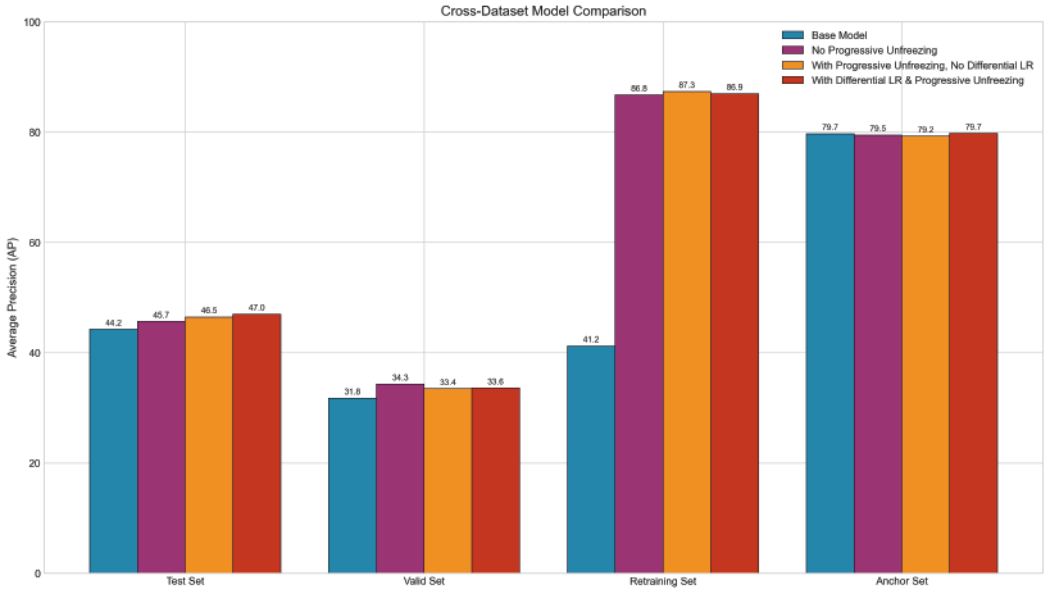


Fig. 57. Cross-Dataset Model Differential LR Comparison Diagram

Figure 57 presents a comparative analysis of the retraining strategies. The implementation of Differential Learning Rate (LR) yields the optimal balance between plasticity and stability. The model achieves superior generalization, reaching a peak Test Set AP of 47.0, which surpasses both the standard progressive unfreezing (46.5) and the baseline (44.2). Crucially, this technique effectively neutralizes the catastrophic forgetting observed in the previous experiment; the Anchor Set performance is fully restored to the baseline value of 79.7 AP, demonstrating that the model can acquire new domain-specific features without degrading its original knowledge base.

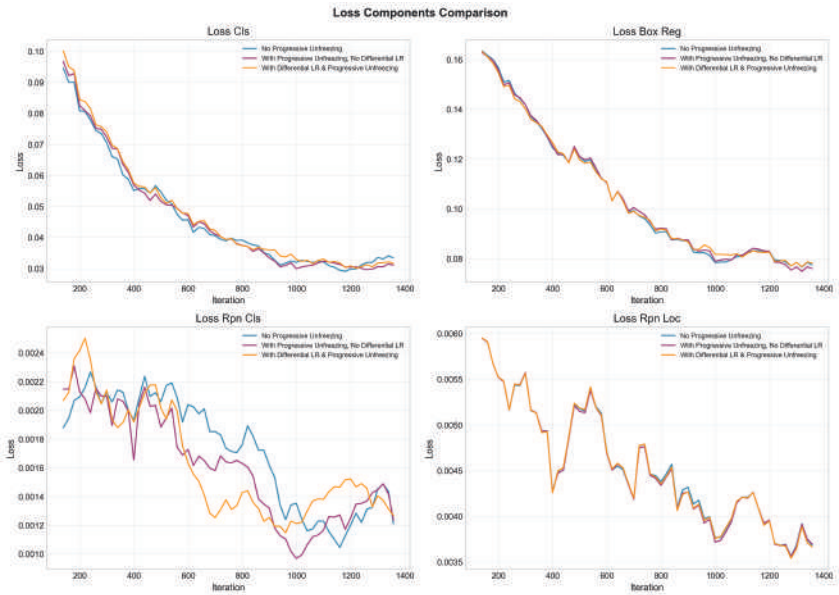


Fig. 58. Progressive Unfreezing Loss Components Comparison Diagram

Figure 58 details the impact of Differential Learning Rate on training convergence. The most significant improvement is evident in the Region Proposal Network Classification (Loss Rpn Cls), where the Differential LR model exhibits a deeper and more stable convergence trajectory (Yellow line) compared to the standard progressive approach. This suggests that by dampening the learning rate for the backbone layers, the RPN is able to fine-tune its objectness criteria more effectively. The reduced variance in the loss curve indicates that the feature extractor remains stable, allowing the head to optimize without battling shifting feature representations.

Dynamic Iteration Scaling

Parameter	HPO-Optimized Value
Max Iterations	1358
Default Dataset Size	126 Images (63 new + 63 anchor at 100% ratio)
IMS PER BATCH	3

$$epochs = \frac{iterations \times images\_per\_batch}{total\_images}$$

$$epochs = \frac{1358 \times 3}{126} \approx 32.3$$

Base Formula

$$iterations = \frac{epochs \times total\_images}{images\_per\_batch}$$

Scaling Formula

To address the potential for over- or under-training caused by fixed iteration counts on varying dataset sizes, the final transfer learning technique applied is Epoch-Based Iteration Scaling. By analyzing the HPO-optimized configuration (1358 iterations, 126 images, batch size 3), a derived baseline training duration of approximately 32.3 epochs is achieved. This value now serves as the normalized constant for the system. Consequently, for all future retraining, the total iterations are dynamically calculated to ensure the model always completes 32 full passes over the data, guaranteeing consistent convergence behavior regardless of the dataset volume.

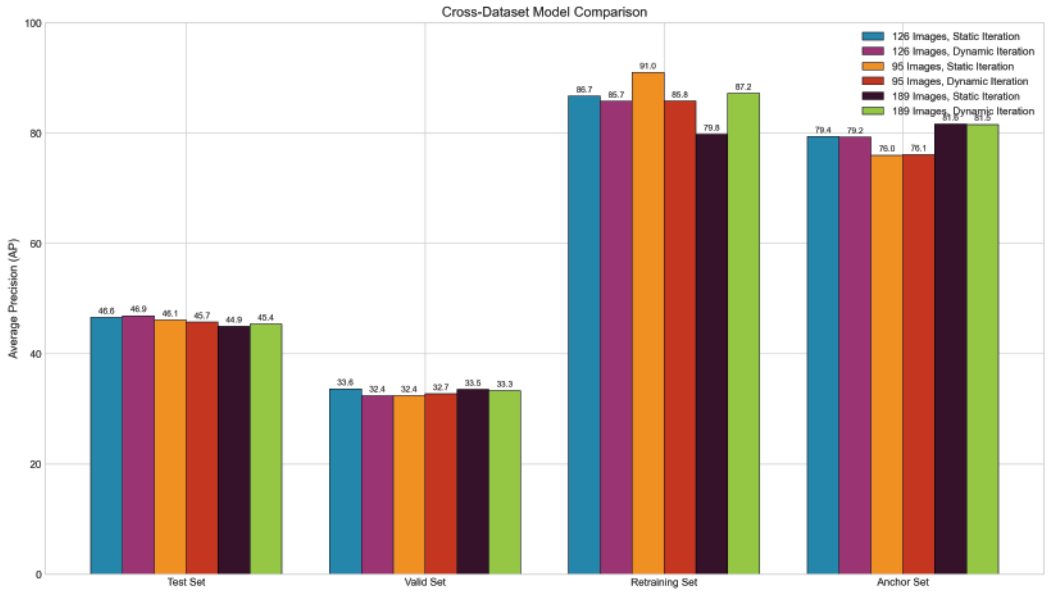


Fig. 59. Cross-Dataset Model Dynamic Iteration Scaling Comparison Diagram

Figure 59 validates the efficacy of Dynamic Iteration Scaling across varying dataset sizes. The most critical finding is observed in the 189-Image subset (Large Dataset). Under the static iteration regime, the model exhibited significant underfitting, achieving a Retraining AP of only 79.8. However, applying dynamic scaling extended the training duration proportionally, resulting in a dramatic performance recovery to 87.2 AP (+7.4 improvement). Conversely, for the 95-image subset (Small Dataset), the static approach yielded an artificially high Retraining AP of 91.0, indicative of overfitting. Dynamic scaling effectively regularized this

behavior by reducing the iteration count, lowering the Retraining AP to a more realistic 85.8 while maintaining stable generalization on the Test Set. This confirms that scaling enables the model to automatically normalize its 'learning exposure' regardless of data volume.

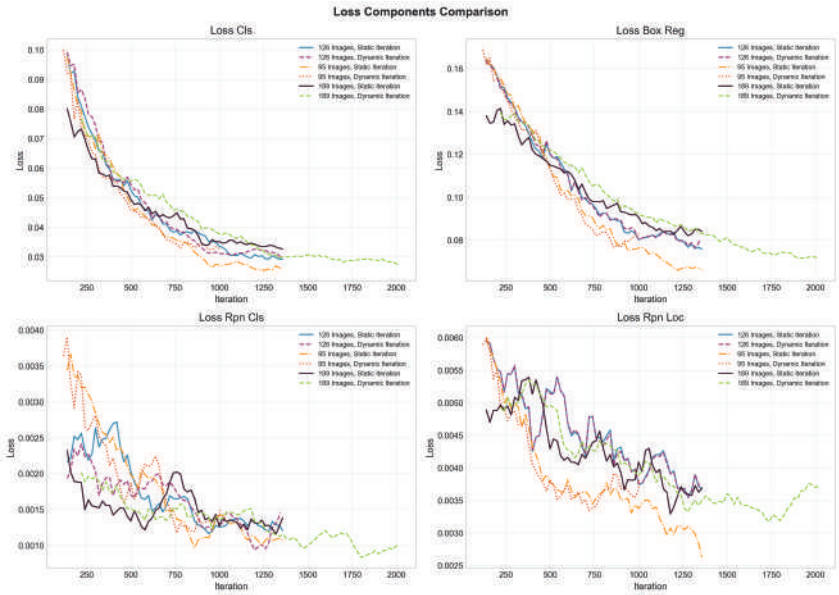


Fig. 60. Dynamic Iteration Scaling Loss Components Comparison Diagram

Figure 60 visualizes the training trajectories under static versus dynamic scaling. The Region Proposal Network Classification (Loss Rpn Cls) graph clearly illustrates the mechanism of the correction. For the 189-Image dataset (Large), the static model (Black line) terminates prematurely at iteration 1358 while the loss is still descending, confirming underfitting. In contrast, the dynamic model (Light Green dashed line) continues optimizing until iteration ≈ 2037, allowing the loss to converge to a significantly lower minimum.Simultaneously, for the 95-Image dataset (Small), the dynamic model (Red dashed line) correctly terminates early at ≈ 1024 iterations, preventing the unnecessary extended training observed in the static model (Yellow dotted line), which contributes to overfitting.

Method	Retraining AP	Test AP	Validation AP	Anchor AP
Baseline	41.19	44.23	31.75	79.68
Progressive Unfreezing	87.30	46.50	33.40	79.20
Differential Learning Rate	–	47.00	–	79.70

Table XVII. Baseline Comparison of Transfer Learning Techniques

Table XVII summarizes the effect of the transfer-learning interventions relative to the baseline model. Progressive Unfreezing produced a large Retraining Set gain (+46.11 AP)

and improved Test AP by +2.27, while maintaining only a small Anchor Set decrease (-0.48 AP). Differential Learning Rate produced the strongest Test Set generalization, increasing Test AP from 44.23 to 47.00 (+2.77 AP), while restoring Anchor AP to approximately the baseline level. These results show that transfer-learning modifications improved adaptation without substantially increasing catastrophic forgetting.

Transfer Learning Techniques Summary

To overcome the performance saturation of hyperparameter tuning, the optimization strategy incorporated three distinct transfer learning interventions. First, Progressive Unfreezing stabilized the training of low-level features, boosting Test AP to 46.5 while mitigating gradient shock in the Region Proposal Network. Subsequently, Differential Learning Rate ('Backbone Chilling') was implemented to balance plasticity and stability; by applying a 0.1x multiplier to the backbone, this technique achieved the highest generalization score (Test AP 47.0) and fully restored the Anchor Set performance (79.7 AP), effectively neutralizing forgetting. Finally, Dynamic Iteration Scaling was introduced to normalize training exposure across varying dataset sizes. This technique proved critical for scalability, correcting significant underfitting in larger datasets (yielding a +7.4 AP recovery) and preventing overfitting in smaller subsets. Collectively, these architectural modifications define a robust, adaptive retraining pipeline capable of maintaining high detection accuracy and stability under dynamic field conditions.

5.3.4 Loss-Based Benchmarking & Hard Example Mining

The transition from AP-based to Loss-based benchmarking was necessitated by the performance saturation observed in earlier experiments. While AP measures the final detection outcome, it often fails to capture subtle improvements in model confidence once a certain accuracy plateau is reached. Consequently, the pivot to Loss-based metrics is driven by the objective to quantify the convergence quality and optimization stability of the training process. This deeper diagnostic view is crucial for fine-tuning, as it identifies how well the model has learned the features, rather than simply how many it detected.

Mining Methodology

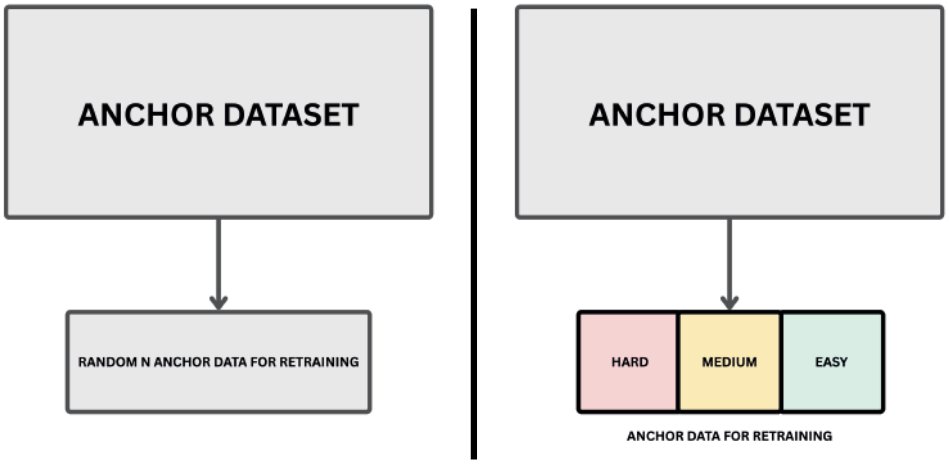


Fig. 61. Anchor Data Selection | Random Selection (Left) Difficulty-Based Selection (Right)

Figure 61 illustrates a critical limitation in the baseline anchor dataset selection methodology. The reliance on random sampling introduces significant stochastic variance, rendering the retraining process 'lottery-based' and degrading experimental determinism. To mitigate this, the proposed solution implements Difficulty-Stratified Sampling. By explicitly categorizing samples into 'Hard,' 'Medium,' and 'Easy' strata prior to selection, this approach eliminates chance as a variable, ensuring that the model is consistently exposed to a balanced and reproducible distribution of difficulty.

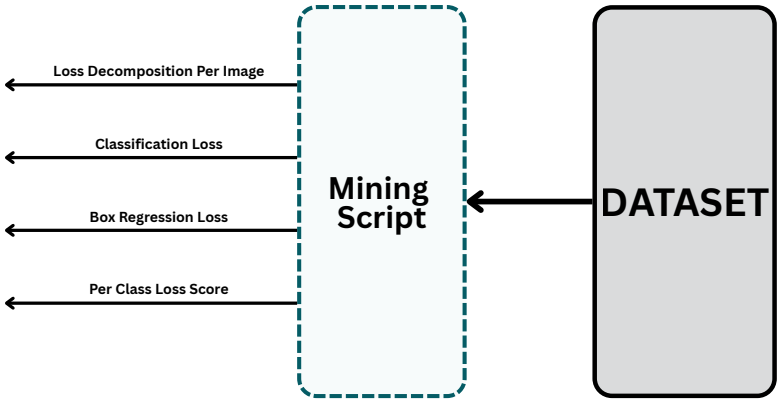


Fig. 62. Mining Script Diagram

To enable granular difficulty classification at the image level, a dedicated Data Mining Script was integrated into the pipeline. As illustrated in Fig. 62, this module systematically

traverses the dataset, executing inference to compute a comprehensive Loss Decomposition for every sample. Specifically, it isolates the Classification Loss (object recognition error) and Box Regression Loss (localization error), alongside a specific Per-Class Loss Score. Furthermore, the script aggregates these metrics to rank images by total loss, effectively stratifying the dataset into difficulty tiers for subsequent visualization and analysis.

Anchor Dataset Classification

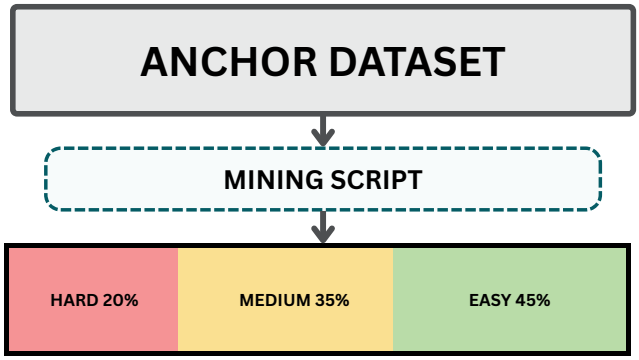


Fig. 63. Anchor Dataset Classification Diagram

The percentile thresholds (80^{th} and 45^{th}) were rigorously selected to balance optimization intensity with model stability. The 80^{th} percentile threshold isolates the top 20% of high-loss samples, aligning with the Pareto Principle[46], which suggests that a minority of difficult cases often account for the majority of model error. Conversely, the 45^{th} percentile boundary preserves the bottom 45% of the dataset, comprising 'easy' and 'stable' examples, to serve as a regularization anchor. This substantial proportion of low-loss data ensures that the model maintains its foundational feature representations, thereby preventing forgetting while the optimizer focuses on the challenging upper quintile.

Difficulty-Based Data Selection

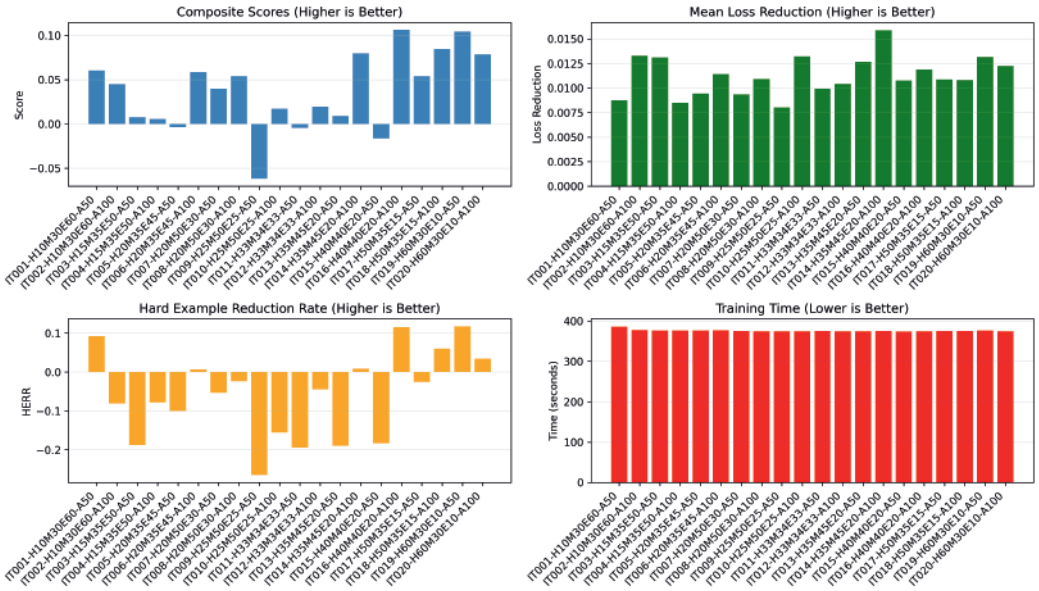


Fig. 64. Difficulty-Based Data Selection Experiments Diagram

To empirically determine the optimal Data Selection Mix, a comprehensive grid search illustrated in Fig. 64, was conducted across 20 distinct model configurations (IT001–IT020). Each iteration varied the distribution of Hard, Medium, and Easy samples to evaluate the trade-off between learning plasticity and model stability. As detailed in the benchmark report, Experiment 16 (IT016) emerged as the global optimum, achieving the highest Composite Score of 0.1062. This configuration is defined by a stratified distribution of 40% Hard, 40% Medium, and 20% Easy samples, utilizing a full anchor ratio of 1.0.

The superiority of the 40-40-20 split suggests that the retraining process benefits most from a high density of challenging and transitional examples. By allocating 80% of the dataset to Hard and Medium cases, the model is forced to adapt to complex gradients and decision boundary edge cases. Crucially, the retention of a 20% 'Easy' anchor prevents the optimization trajectory from diverging, ensuring that foundational feature representations remain stable. This configuration narrowly outperformed the runner-up (IT019), demonstrating that a balanced approach, incorporating a significant portion of 'Medium' difficulty samples, yields better overall optimization than strategies that focus exclusively on the hardest outliers.

Data Filtration

To further enhance training efficacy, a data filtering module was integrated into the preliminary stage of the retraining pipeline. The primary objective of this intervention is to prune 'easy' (low-loss) samples, effectively increasing the dataset's mean loss to prioritize high-information examples. This selection strategy is governed by a hybrid threshold formula, defined as:

Hybrid Threshold Formula

$$threshold = max(anchor\_threshold, self\_threshold)$$

Where:

$$anchor\_threshold = \mu_{anchor} + (K \times 2 \times \sigma_{anchor})$$

μ_{anchor} = mean loss of anchor dataset
σ_{anchor} = standard deviation of anchor losses
K = filtering strictness

$$self\_threshold = P_K(new\_losses)$$

Where P_K is the percentile of new data losses calculated as:

$$percentile = high\_pct - K \times (high\_pct - low\_pct)$$

With defaults: high_pct = 80, low_pct = 45

A sample is kept if

$$loss_i \leq threshold$$

A hybrid filtering strategy that integrates both anchor-relative and self-relative thresholds to govern data selection is implemented. The anchor-relative component enforces consistency, ensuring that filtered samples align with the model's established baseline of difficulty. Complementing this, the self-relative component provides adaptive capability, dynamically adjusting to distributional shifts within the incoming data stream. By adopting the maximum of these two values as the effective threshold, the system achieves operational robustness: it prevents over-aggressive filtering when the distributions diverge, while maintaining a strict quality floor when the incoming data resembles the anchor distribution.

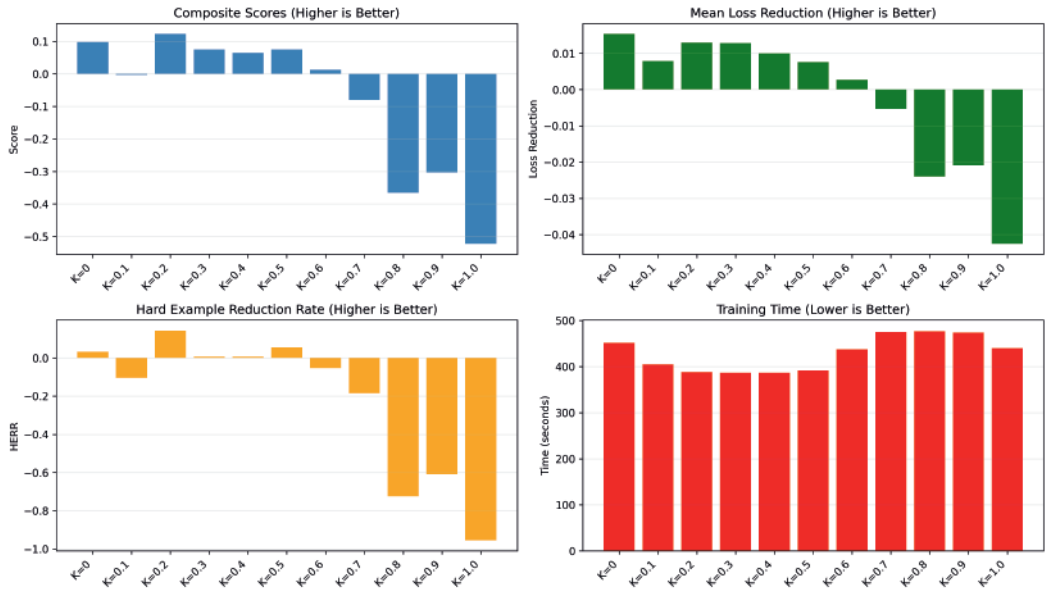
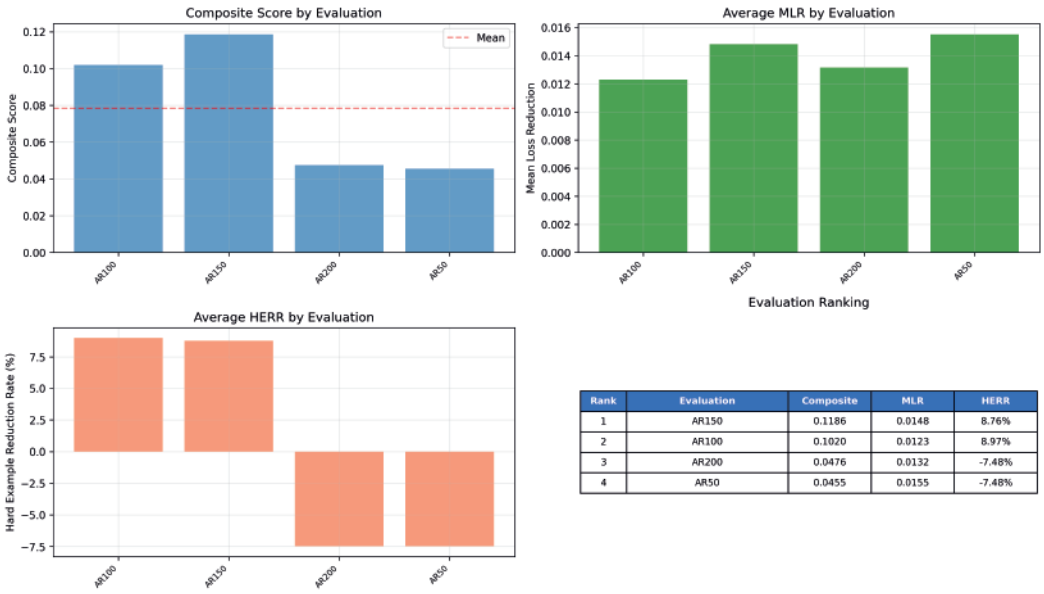


Fig. 65. Data Filtering Strictness Experiments Diagram

Figure 65 identifies $K = 0.2$ as the optimal strictness threshold for the data filtering algorithm. As illustrated in the Composite Scores graph, this configuration achieves the global maximum, outperforming the unfiltered baseline ($K = 0$) and all stricter settings. This value represents the ideal equilibrium between quality and quantity: the filter is sufficiently strict to remove detrimental outliers (noise or mislabeled data) that confuse the training process, yet lenient enough to preserve the diverse examples necessary for generalization. As the strictness factor (K) increases beyond 0.2, performance metrics begin to degrade, confirming that overly aggressive filtering inadvertently discards valuable training data.

The consequences of excessive strictness are further evidenced by the Mean Loss Reduction (MLR) and Hard Example Reduction Rate (HERR) metrics. While $K = 0.2$ yields the highest positive HERR, indicating effective resolution of difficult cases, higher K values (> 0.6) result in deeply negative scores. This precipitous drop demonstrates that as the filter becomes stricter and removes a larger volume of images, the model suffers from information starvation. By systematically culling too many samples, the model loses the critical mass of data required to define complex class boundaries, ultimately leading to a regression in detection capabilities.

Anchor Ratio



Rank	Evaluation	Composite	MLR	HERR
1	AR150	0.1186	0.0148	8.76%
2	AR100	0.1020	0.0123	8.97%
3	AR200	0.0476	0.0132	-7.48%
4	AR50	0.0455	0.0155	-7.48%

Fig. 66. Anchor Ratio Experiments Diagram

Figure 66 illustrates the non-linear relationship between anchor data volume and retraining performance. The results clearly indicate that moderate anchor ratios—specifically 150% (AR150) and 100% (AR100)—yield superior outcomes, with AR150 achieving the highest Composite Score of 0.1186. In contrast, extreme ratios resulted in significant performance degradation: the 50% ratio (AR50) likely provided insufficient stability against catastrophic forgetting, while the 200% ratio (AR200) appears to have diluted the learning signal from the new data. Notably, both AR50 and AR200 exhibited negative Hard Example Reduction Rates ($\approx -7.48\%$), confirming their inability to effectively resolve difficult samples.

Loss-Based Benchmarking Summary

Component	Value
Anchor Dataset Classification	Hard = >80th | Med = 45th - 80th | Easy =<45th
Data Selection Mix	Hard = 40% | Med = 40% | Easy = 20%
K (Strictness)	0.2
Anchor Ratio	150%

To address the stochastic instability inherent in random sampling, the retraining pipeline replaced "lottery-based" selection with a deterministic Difficulty-Stratified Sampling method-

ology. Enabled by a custom Data Mining Script that decomposes loss metrics into classification and localization errors, the system categorizes data into distinct difficulty strata based on the Pareto Principle (80^{th} percentile for Hard) and stability requirements (45^{th} percentile for Easy). Grid search optimization subsequently identified Configuration IT016 as the global optimum (Composite Score: 0.1062), utilizing a 40% Hard, 40% Medium, and 20% Easy distribution. This ratio maximizes learning plasticity through high-difficulty exposure while preserving a critical 20% stability anchor to prevent optimization divergence.

Complementing this stratification, the pipeline incorporates a Hybrid Threshold Filtering module to prune low-information samples. Optimization experiments determined that a strictness factor of $K = 0.2$ achieves the ideal equilibrium, removing detrimental label noise without inducing the "information starvation" observed at higher strictness levels ($K > 0.6$). Finally, the analysis of anchor data volume established an Anchor Ratio of 150% (AR150) as the superior configuration (Score: 0.1186). This moderate ratio effectively balances the retention of prior knowledge against the integration of new features, outperforming the instability of lower ratios and the signal dilution associated with excessive anchor data.

5.3.5 Best Results

Component	Value
Base Learning Rate	0.0000151
Max Iterations	Dynamic | Epoch = 32
Weight Decay	0.000363
Warmup Iterations	196
Warmup Factor	0.00181
Ims Per Batch	3
ROI Batch Per Image	192
Step 1 Ratio	Dynamic
Step 2 Ratio	Dynamic
LR Schedule Steps	Dynamic
Seed	2025
Progressive Unfreezing	stem - res2 = frozen
Differential Learning Rate	res3- res 5 = 0.1 x Base LR
Dynamic Iteration Scaling	Enabled
Anchor Dataset Classification	Hard = >80th | Med = 45th - 80th | Easy =<45th
Data Selection Mix	Hard = 40% | Med = 40% | Easy = 20%
K (Strictness)	0.2
Anchor Ratio	150%

Table XVIII. Optimal Specifications for Retraining

Table XVIII synthesizes the optimal retraining specifications for the C.Scope.AI system, representing the culmination of an extensive iterative optimization process. This configuration integrates the global optimums identified across all experimental phases: the Trial 20 hyperparameters (Base LR 1.51×10^{-5}, Seed 2025), the architectural transfer learning techniques (Freezing stem-res2, Chilling res3-res5), and the data selection strategies (40-40-20 Mix, 150% Anchor Ratio). Collectively, these parameters define the stable, reproducible, and robust state of the C.Scope.AI V2 retraining module.

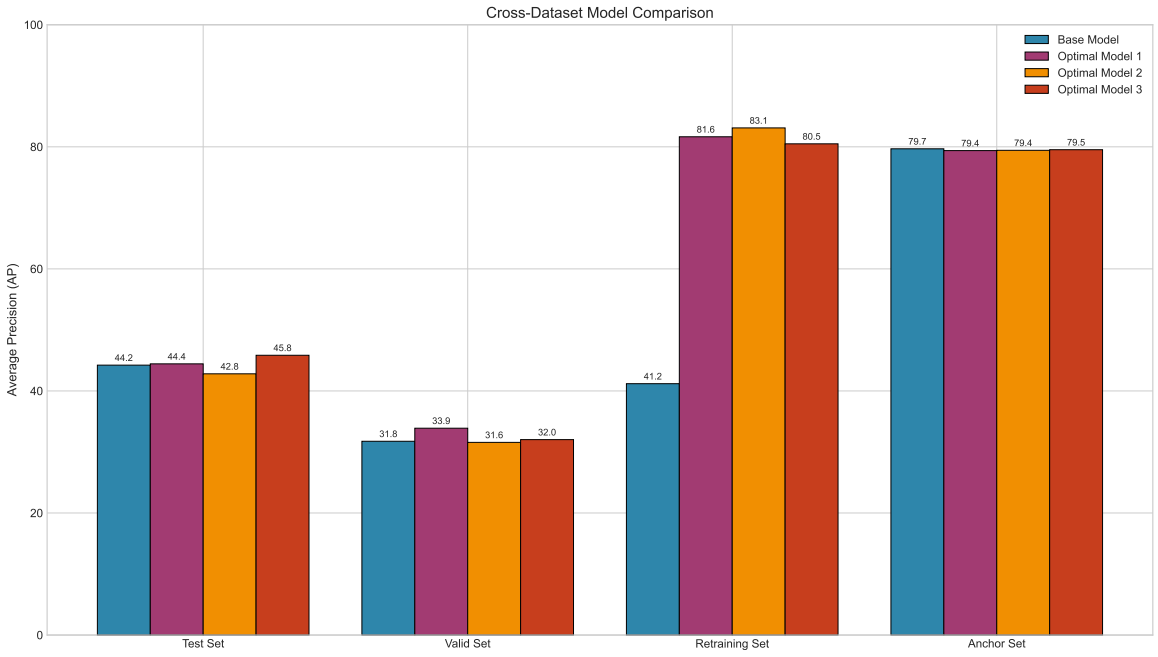


Fig. 67. Base Model vs Optimal Model Cross-Dataset Comparison Diagram

Based on the comparative analysis presented in Fig 67, the retraining pipeline demonstrates a decisive performance improvement over the Base Model. The most significant gain is observed in the Retraining Set, where the optimized models achieved a peak AP of 83.10 (Optimal Model 2), effectively doubling the Base Model's performance of 41.19. This drastic increase confirms the system's ability to rapidly adapt to new, domain-specific data. Crucially, this adaptation successfully generalized to unseen data, with Optimal Model 3 securing the highest Test Set AP of 45.84, surpassing the Base Model (44.23) and validating the robustness of the transfer learning strategies.

Furthermore, the results confirm the effectiveness of the stability preservation mechanisms (such as the 150% Anchor Ratio). Performance on the Anchor Set remained virtually unchanged, with the retrained models (e.g., Optimal Model 3 at 79.53 AP) showing negligible deviation from the Base Model (79.68 AP), indicating that the system successfully integrated new features without suffering from forgetting.

Using the baseline AP values established in Tables X–XIII, Table XIX summarizes the peak AP gains produced by the optimized retraining experiments.

Dataset	Baseline AP	Peak Optimized AP	AP Gain	Relative Change
Anchor	79.68	79.53	-0.15	-0.19%
Retraining	41.19	83.10	+41.91	+101.75%
Test	44.23	45.84	+1.61	+3.64%
Validation	31.75	33.90	+2.15	+6.77%

Table XIX. Baseline and Peak AP Comparison Across Datasets

Table XIX shows that the largest improvement occurred on the Retraining Set, while the Anchor Set remained effectively stable. This confirms that the pipeline improved adaptation to new data without producing meaningful catastrophic forgetting.

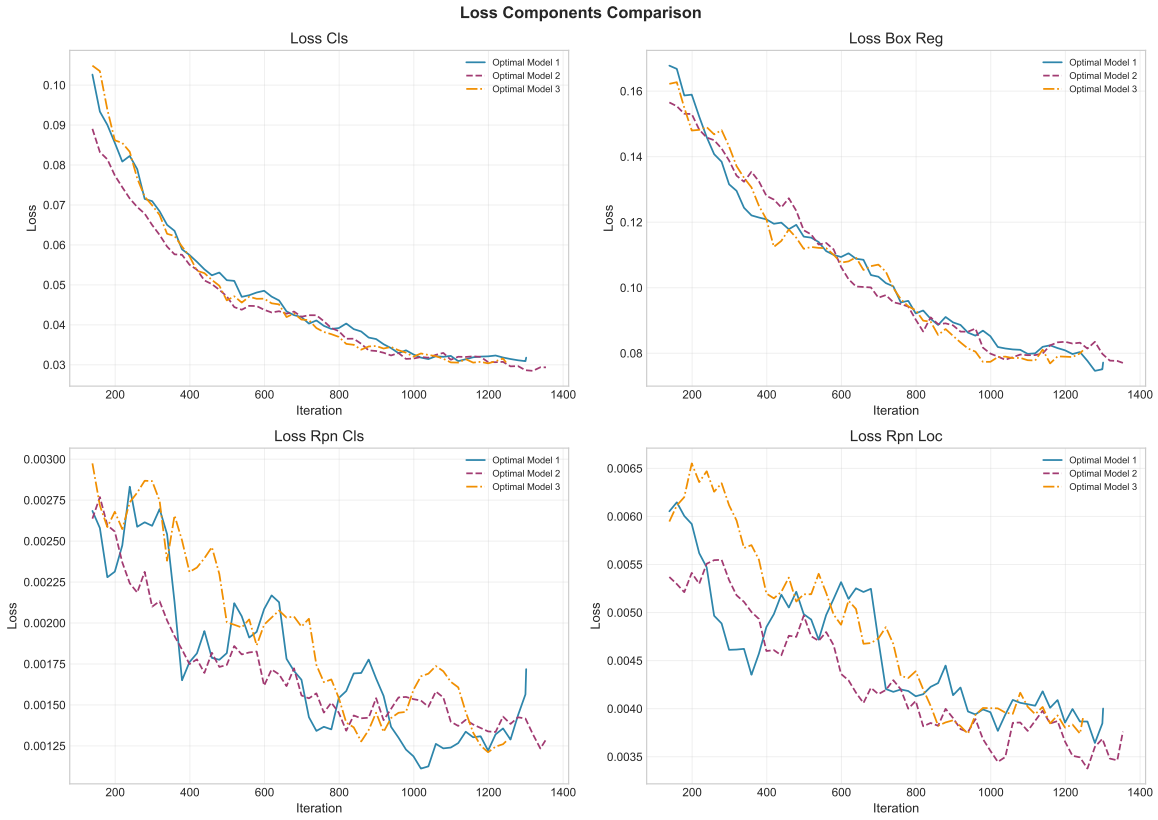


Fig. 68. Optimal Models Loss Components Comparison Diagram

Figure 68 illustrates the training convergence of the three top-performing models initialized with the optimal specifications defined in Table XVIII. Across the primary detection heads—Classification Loss and Box Regression Loss—all three variants exhibit tightly clustered, monotonic trajectories, confirming the high reproducibility of the retraining pipeline. This stability validates the effectiveness of the architectural interventions (Backbone Chilling

and Progressive Unfreezing), as the models consistently minimize error without the volatility or divergence often associated with retraining on small, difficult datasets.

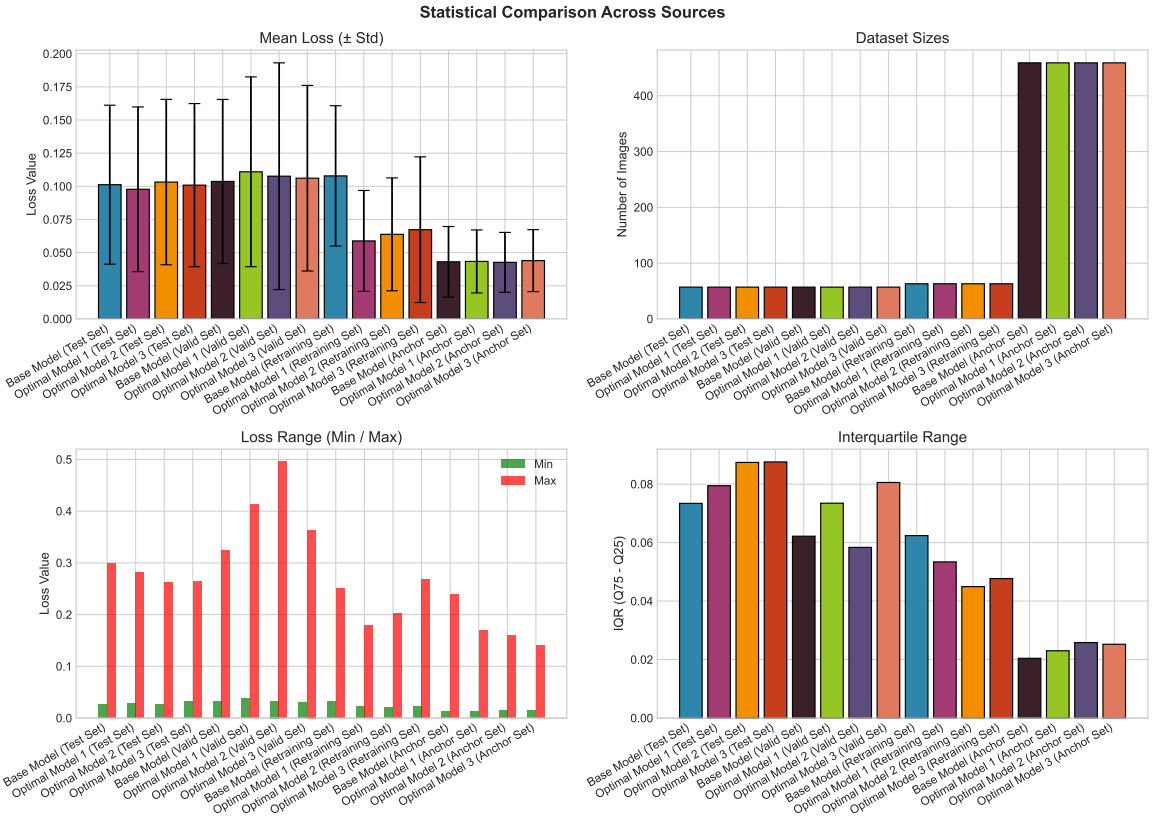


Fig. 69. Base vs Optimal Models Loss Charts Comparison

Figure 69 corroborates the effectiveness of the retraining pipeline through a statistical loss analysis across four datasets. The most profound impact is observed in the Retraining Set, where the Base Model exhibited high variance and high mean loss (≈ 0.175), indicating significant difficulty in processing the new domain data. The Optimal Models drastically corrected this, compressing the variance and reducing mean loss to ≈ 0.05, quantitatively proving that the system maximized plasticity by transforming difficult samples into well-learned features. Simultaneously, the Anchor Set metrics validate the stability preservation mechanisms; the Optimal Models show only a negligible deviation (< 0.025) from the Base Model's near-zero loss, confirming that catastrophic forgetting was effectively neutralized while maintaining robust generalization on unseen Test and Validation data.

Fig. 70. Base vs Optimal Models Heatmap Comparison

Figure 70 provides a granular, image-level validation of the retraining efficacy. The heatmap reveals a massive inversion in the Retraining Set difficulty profile, where the dense "Hard" (Red) bands observed in the Base Model are almost entirely transformed into "Easy" (Green) regions in the Optimal Models. This visual shift quantitatively proves that the system successfully maximized plasticity, converting previously challenging domain-specific samples into high-confidence detections. Simultaneously, the Anchor Set columns display a preserved "Easy" difficulty distribution across all models, confirming that the architectural interventions effectively prevented the introduction of new errors or forgetting during the adaptation process.

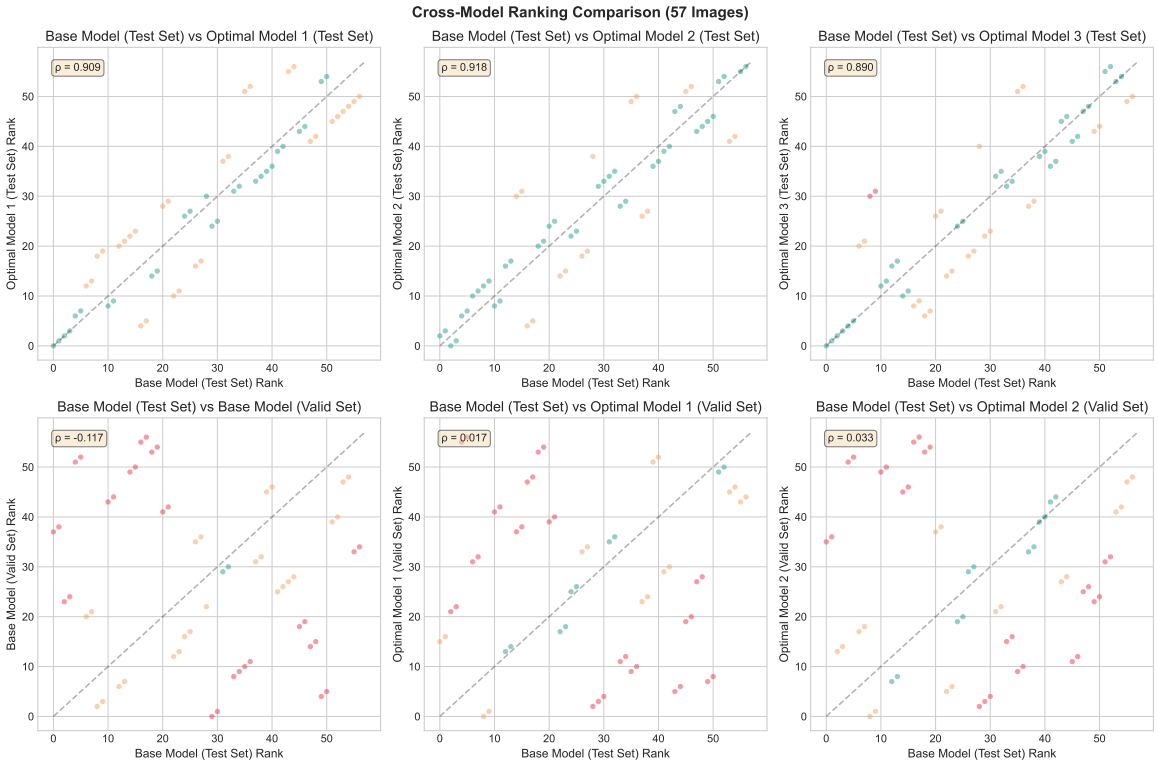


Fig. 71. Base vs Optimal Models Ranking Scatter Comparison

Figure 71 validates the stability of the optimization pipeline through a rank-correlation analysis. The top row reveals a near-perfect linear relationship ($\rho > 0.89$) between the Base Model and all three Optimal Models on the Test Set, demonstrating that the hierarchy of image difficulty remained preserved throughout the retraining process. This consistency confirms that the architectural interventions enhanced the model's detection capabilities uniformly without disrupting its fundamental understanding of visual complexity. Consequently, the retrained models exhibit predictable behavior, systematically resolving harder images while retaining the same relative assessment of data quality as the baseline.

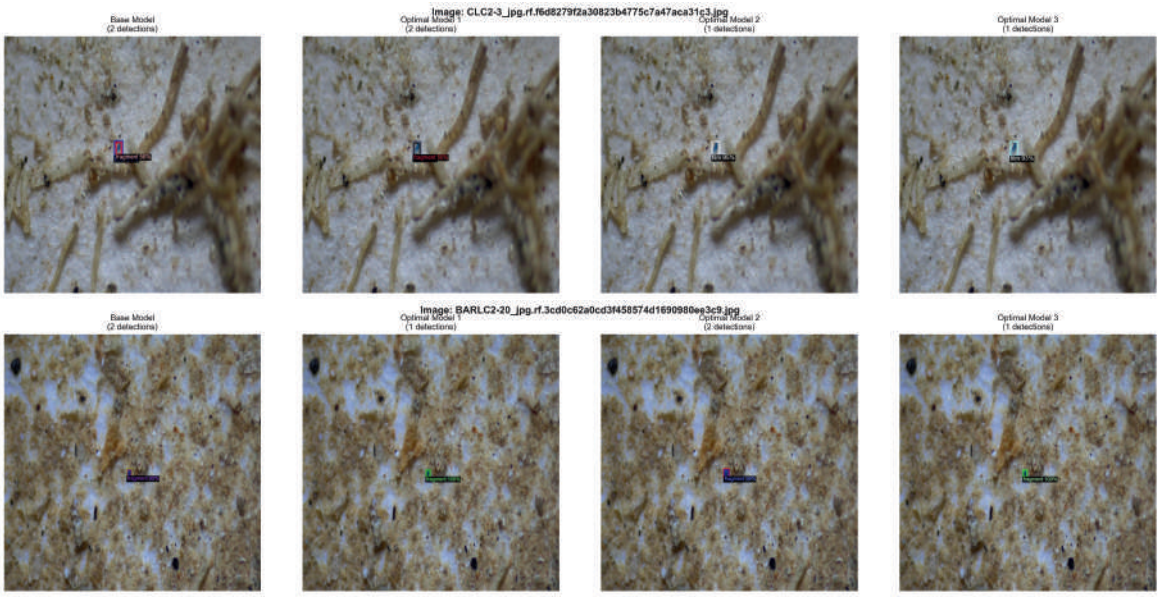


Fig. 72. Base vs Optimal Models Real World Detection Comparison

Figure 72 provides qualitative evidence of the system's enhanced sensitivity to small-scale microplastics within unseen validation and test sets. While the Base Model exhibits significant ambiguity, often yielding lower confidence scores (e.g., 56%) or failing to resolve fine-grained textures, the Optimal Models demonstrate decisive improvements. As seen in the comparison, Optimal Models 2 and 3 not only correct misclassifications but also lock onto minute targets with high confidence ($> 90\%$).

This capability is critical for field deployment, where "small objects" constitute the majority of elusive microplastic particles that standard models typically overlook. This superior performance on small objects is directly attributable to the optimized specifications detailed in Table XVIII. The configuration utilizes a high ROI Batch Per Image (192), which forces the Region Proposal Network to sample a denser array of candidate boxes, significantly increasing the probability of capturing small features. Furthermore, the application of Differential Learning Rate on the backbone ensures that the high-resolution feature maps, essential for defining the edges of small objects, are fine-tuned without being distorted by gradient noise. This synergy between dense sampling and stabilized feature extraction allows the retrained models to "substantially outperform" the baseline in recovering small, low-pixel-count targets.

5.3.6 Best Results Summary

The final results demonstrate that the C.Scope.AI V2 retraining pipeline achieves a balanced solution to the plasticity-stability problem in local model adaptation. The optimized configuration is not the result of a single parameter change, but the combined effect of the major experimental findings established throughout Section 5.3. As shown in Fig. 50, Base Learning Rate was the dominant hyperparameter affecting Average Precision, which justified the conservative Trial 20 setting summarized in Table XV. This was then strengthened by the transfer-learning and data-selection strategies summarized in Table XVIII, including frozen stem-res2 layers, chilled res3-res5 backbone updates, dynamic iteration scaling, 40-40-20 difficulty-based sampling, $K = 0.2$ filtering, and a 150% anchor ratio. Together, these settings define the final retraining pipeline as a controlled adaptation strategy rather than a purely trial-based optimization result.

As summarized in Table XIX, the strongest evidence of plasticity is the Retraining Set improvement. Relative to the baseline AP in Table XI, the peak optimized result increased AP from 41.19 to 83.10, a +41.91 AP gain or approximately 101% relative improvement. This improvement is supported by the loss-based evidence in Fig. 69 and Fig. 70, where the mean retraining loss decreased from approximately 0.175 to 0.05 and the heatmap shifted from dense hard-example regions to predominantly easy detections. Therefore, the gain is not only a numerical AP increase; it demonstrates that the retraining pipeline converted previously difficult domain-specific samples into learned features.

Equally important, this plasticity was achieved without meaningful catastrophic forgetting. The Anchor Set AP remained nearly unchanged, decreasing only from 79.68 to 79.53, which preserves approximately 99.8% of the original detection capacity. The Test Set also improved from 44.23 AP to 45.84 AP, while the Validation Set improved from 31.75 AP to approximately 33.9 AP. These gains are smaller than the Retraining Set improvement, but they are critical because they show that the adapted model did not simply memorize the retraining data. The rank-correlation analysis in Fig. 71 further supports this conclusion, showing that the optimized models preserved the baseline ordering of image difficulty ($\rho > 0.89$). This indicates that the model improved detection performance while maintaining a stable internal representation of visual complexity.

Finally, the qualitative comparison in Fig. 72 demonstrates the practical value of the optimized pipeline. The retrained models show stronger confidence and localization on small, low-pixel-count microplastic targets, which are among the most difficult cases for

field deployment. Overall, the final result confirms that the proposed retraining feature is not merely functional as a software module; it also produces measurable model improvement, preserves prior knowledge, and supports reliable local adaptation under the human-in-the-loop workflow required by C.Scope.AI V2.

Chapter 6

Conclusions

6.1 Summary

In summary, research presents the comprehensive optimization of the established automated imaging system designed to address the issues of manual microplastic microscopy of Version 1. The system re-engineering encompassed both hardware and software architectures, prioritizing end-user accessibility, system autonomy, and adaptive learning capabilities.

The notable hardware modification involved the transition to an H-Bot kinematic configuration for the XY Table. Constructed primarily from 3D-printed components, the mechanism utilizes a GT2 timing belt routed through a strategic pulley arrangement driven by NEMA 17 stepper motors to govern X and Y-axis actuation. While core electronic components from the first iteration, including motor drivers, the CNC shield, and the microcontroller, were retained to maximize resource efficiency, the mechanical redesign yielded significant physical optimizations. Simultaneously, the software architecture underwent significant refactoring to eliminate reliance on external dependencies, shifting model storage to a local environment within the application. This architectural change simplifies the installation and setup process, ensuring accessibility for non-technical users. A critical functional addition is the Retraining Pipeline, which comprises six distinct stages: Environment & Model Setup, Dataset Mining, Data Filtering, Data Combination, Training Data Preparation, and Training & Evaluation. To facilitate this, an Annotation Window was developed for pre-retraining data labeling, while a Summary Window grants users final authority over model deployment post-retraining. Unit Testing is observed to ensure modular functionality. Furthermore, the system underwent evaluation by the Biology Department of the University of San Carlos and added feasible changes for their feedback. The user-friendly interface, guided by a comprehensive user guide, enhances ease of navigation and learning.

Quantitative results demonstrate the efficacy of these improvements. The hardware redesign achieved a 42% reduction in footprint area and a 55% reduction in the height profile compared to Version 1, while reducing the unit cost from approximately PHP 14,500 to PHP 8,934. Software benchmarking indicated a significant performance increase and stability, registering a 9.80% and 11.71% speedup in detection time for binary and multiclass models, respectively, alongside a reduction in memory utilization of approximately 1.5 GB. Finally, the retraining pipeline increases the average precision of a model by **5%-8%** from 60 new data, plus an 85% relative improvement while maintaining 97% of its original detection capacity.

In conclusion, the enhancement of the automated imaging system not only resolves the inefficiencies of manual microplastic determination but also confirms its readiness for mass deployment. Through the strategic integration of a cost-effective hardware design and a standalone, adaptive software architecture, the system ensures both operational accessibility and long-term viability. Ultimately, this research delivers a scalable, standardized solution capable of bridging the gap between advanced computer vision and practical ecological monitoring.

6.2 Recommendations

The researchers propose the following optimization for the system's further development. First, they suggest on leveraging CUDA-enabled/ROCm-enabled GPU acceleration to optimize computational performance, moving beyond the constraints of the current CPU-exclusive architecture. Secondly, we recommend the adoption of state-of-the-art CNN architectures to supersede the current framework. Furthermore, future iterations should explore the integration of Small Language Models (SLMs) or heavily quantized open-source Large Language Models (LLMs). These advanced language models can be leveraged to augment process automation and refine classification logic, potentially enabling a multimodal approach to microplastic analysis. Third, the classification scope of the Machine Learning Model should be expanded to encompass a broader range of microscopic entities beyond microplastics. Future iterations should incorporate classes for common non-target artifacts, or implement more classes of microplastics such as pellets and foams.

References

[1] I. Napper and R. Thompson, "Plastics and the environment," *Annual Review of Environment and Resources*, vol. 48, no. Volume 48, 2023, pp. 55–79, 2023. [Online]. Available: https://www.annualreviews.org/content/journals/10.1146/annurev-environ-112522-072642

[2] M. Go, A. Ybañez, A. Illano, F. Cababat, and L. De La Calzada, "Microplastics in beach sediments, seawater, and common fish in tourist destinations." *Global Journal of Environmental Science & Management (GJESM)*, vol. 10, no. 4, 2024.

[3] T. van Emmerik, "Macroplastic research in an era of microplastic," pp. 1–2, 2021.

[4] L. C. De Sá, M. Oliveira, F. Ribeiro, T. L. Rocha, and M. N. Futter, "Studies of the effects of microplastics on aquatic organisms: what do we know and where should we focus our efforts in the future?" *Science of the total environment*, vol. 645, pp. 1029–1039, 2018.

[5] M. S. Bhuyan, "Effects of microplastics on fish and in human health," *Frontiers in Environmental Science*, vol. 10, p. 827289, 2022.

[6] Y. Li, L. Tao, Q. Wang, F. Wang, G. Li, and M. Song, "Potential health impact of microplastics: a review of environmental distribution, human exposure, and toxic effects," *Environment & Health*, vol. 1, no. 4, pp. 249–257, 2023.

[7] SEAMAP-ASEAN, "SEAMAP-ASEAN: Southeast Asia Marine Protected Areas," 2025, accessed: 2025-04-07. [Online]. Available: https://seamap-asean.org/

[8] K. Blackburn and D. Green, "The potential effects of microplastics on human health: What is known and what is unknown," *Ambio*, vol. 51, no. 3, pp. 518–530, 2022.

[9] N. N. Phuong, A. Zalouk-Vergnoux, L. Poirier, A. Kamari, A. Châtel, C. Mouneyrac, and F. Lagarde, "Is there any consistency between the microplastics found in the field and those used in laboratory experiments?" *Environmental pollution*, vol. 211, pp. 111–123, 2016.

[10] C. Wu, K. Zhang, and X. Xiong, "Microplastic pollution in inland waters focusing on asia," *Freshwater microplastics: Emerging environmental contaminants?*, pp. 85–99, 2018.

[11] H. T. Nair and S. Perumal, "Trophic transfer and accumulation of microplastics in freshwater ecosystem: risk to food security and human health," *International Journal of Ecology*, vol. 2022, no. 1, p. 1234078, 2022.

[12] L. A. Bucol, E. F. Romano, S. M. Cabcaban, L. M. D. Siplon, G. C. Madrid, A. A. Bucol, and B. Polidoro, "Microplastics in marine sediments and rabbitfish (siganus fuscescens) from selected coastal areas of negros oriental, philippines," *Marine Pollution Bulletin*, vol. 150, p. 110685, 2020.

[13] N. O. US Department of Commerce and A. Administration, "A guide to plastic in the ocean," *NOAA's National Ocean Service*, Sep 2018. [Online]. Available: https://oceanservice.noaa.gov/hazards/marinedebris/plastics-in-the-ocean.html

[14] C. Scopetani, M. Esterhuizen-Londt, D. Chelazzi, A. Cincinelli, H. Setälä, and S. Pflugmacher, "Self-contamination from clothing in microplastics research," *Ecotoxicology and environmental safety*, vol. 189, p. 110036, 2020.

[15] T. Thammasanya, S. Patiam, E. Rodcharoen, and P. Chotikarn, "A new approach to classifying polymer type of microplastics based on faster-rcnn-fpn and spectroscopic imagery under ultraviolet light," *Scientific reports*, vol. 14, no. 1, p. 3529, 2024.

[16] M. G. Löder and G. Gerdts, "Methodology used for the detection and identification of microplastics—a critical appraisal," *Marine anthropogenic litter*, pp. 201–227, 2015.

[17] A. Gauci, A. Deidun, J. Montebello, J. Abela, and F. Galgani, "Automating the characterisation of beach microplastics through the application of image analyses," *Ocean & Coastal Management*, vol. 182, p. 104950, 2019.

[18] N. Meyers, A. I. Catarino, A. M. Declercq, A. Brenan, L. Devriese, M. Vandegehuchte, B. De Witte, C. Janssen, and G. Everaert, "Microplastic detection and identification by nile red staining: Towards a semi-automated, cost-and time-effective technique," *Science of the Total Environment*, vol. 823, p. 153441, 2022.

[19] J. A. Bernales, J. K. Lastre, P. J. Toral, P. V. B. Astillo, and J. L. G. S. Cañete, "Development of an imaging system for microplastics sample identification using convolutional neural network," 2023, journal/conference details not available.

[20] E. George, "Occupational hazard for pathologists: microscope use and musculoskeletal disorders," *American journal of clinical pathology*, vol. 133, no. 4, pp. 543–548, 2010.

[21] J. Kristiansen, L. Mathiesen, P. K. Nielsen, Å. M. Hansen, H. Shibuya, H. M. Petersen, S. P. Lund, J. Skotte, M. B. Jørgensen, and K. Søgaard, "Stress reactions to cognitively demanding tasks and open-plan office noise," *International Archives of Occupational and Environmental Health*, vol. 82, pp. 631–641, 2009.

[22] G. Jain and P. Shetty, "Occupational concerns associated with regular use of microscope," *International journal of occupational medicine and environmental health*, vol. 27, pp. 591–598, 2014.

[23] A. Vijendren, G. Devereux, B. Kenway, K. Duffield, V. Van Rompaey, P. Van de Heyning, and M. Yung, "Effects of prolonged microscopic work on neck and back strain amongst male ent clinicians and the benefits of a prototype postural support chair," *International Journal of Occupational Safety and Ergonomics*, 2019.

[24] M. Helander, E. Grossmith, and P. Prabhu, "Planning and implementation of microscope work," *Applied ergonomics*, vol. 22, no. 1, pp. 36–42, 1991.

[25] J. T. Emanuel and R. J. Glonek, "Ergonomic approach to productivity improvement for microscope work," in *Proc AIIE Systems Engineering Conf. Institute for Industrial Engineers, Norcross, GA*, 1976.

[26] I. Söderberg, B. Calissendorff, S. Elofsson, B. Knave, and K. Nyman, "Investigation of visual strain experienced by microscope operators at an electronics plant," *Applied ergonomics*, vol. 14, no. 4, pp. 297–305, 1983.

[27] V. Bhatnager, C. G. Drury, and S. Schiro, "Posture, postural discomfort, and performance," *Human factors*, vol. 27, no. 2, pp. 189–199, 1985.

[28] E. Roudi and S. A. Zakerian, "Evaluating the microscope users occupational health status considering musculoskeletal disorders and visual fatigue at tehran university of medical sciences," *International Journal of Occupational Hygiene*, vol. 11, no. 4, pp. 259–270, 2019.

[29] K. O'shea and R. Nash, "An introduction to convolutional neural networks," *arXiv preprint arXiv:1511.08458*, 2015.

[30] K. Fukushima, "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," *Biological cybernetics*, vol. 36, no. 4, pp. 193–202, 1980.

[31] D. H. Hubel and T. N. Wiesel, "Receptive fields and functional architecture of monkey striate cortex," *The Journal of physiology*, vol. 195, no. 1, pp. 215–243, 1968.

[32] R. Yamashita, M. Nishio, R. K. G. Do, and K. Togashi, "Convolutional neural networks: an overview and application in radiology," *Insights into imaging*, vol. 9, pp. 611–629, 2018.

[33] W. A. Williams, A. Arunprasad, and S. Aravamudhan, "Application of a modified set of googlenet and resnet-18 convolutional neural networks towards the identification of environmentally derived-mpls in the yadkin-pee dee river basin," *Environmental Systems Research*, vol. 13, no. 1, p. 53, 2024.

[34] J. S. Hanvey, P. J. Lewis, J. L. Lavers, N. D. Crosbie, K. Pozo, and B. O. Clarke, "A review of analytical techniques for quantifying microplastics in sediments," *Analytical Methods*, vol. 9, no. 9, pp. 1369–1383, 2017.

[35] M. Yurtsever and U. Yurtsever, "Use of a convolutional neural network for the classification of microbeads in urban wastewater," *Chemosphere*, vol. 216, pp. 271–280, 2019.

[36] L. Ren, S. Liu, S. Huang, Q. Wang, Y. Lu, J. Song, and J. Guo, "Identification of microplastics using a convolutional neural network based on micro-raman spectroscopy," *Talanta*, vol. 260, p. 124611, 2023.

[37] J. Wu, "Introduction to convolutional neural networks," *National Key Lab for Novel Software Technology. Nanjing University. China*, vol. 5, no. 23, p. 495, 2017.

[38] G. Zeng, Y. Ma, M. Du, T. Chen, L. Lin, M. Dai, H. Luo, L. Hu, Q. Zhou, and X. Pan, "Deep convolutional neural networks for aged microplastics identification by fourier transform infrared spectra classification," *Science of The Total Environment*, vol. 913, p. 169623, 2024.

[39] W. Zhang, W. Feng, Z. Cai, H. Wang, Q. Yan, and Q. Wang, "A deep one-dimensional convolutional neural network for microplastics classification using raman spectroscopy," *Vibrational Spectroscopy*, vol. 124, p. 103487, 2023.

[40] J. Lorenzo-Navarro, M. Castrillón-Santana, E. Sánchez-Nielsen, B. Zarco, A. Herrera, I. Martínez, and M. Gómez, "Deep learning approach for automatic microplastics counting and classification," *Science of the Total Environment*, vol. 765, p. 142728, 2021.

[41] A. Kavikondala, V. Muppalla, D. Prakasha, and V. Acharya, "Automated retraining of machine learning models," *International Journal of Innovative Technology and Exploring Engineering*, vol. 8, no. 12, pp. 445–452, 2019.

[42] AutoML Group, "Automl: Automated machine learning," https://www.automl.org/automl/, 2024, accessed: 2025-04-07.

[43] Epistasis Lab, "Tpot: Tree-based pipeline optimization tool," https://epistasislab.github.io/tpot/latest/, 2024, accessed: 2025-04-07.

[44] N. Jayram and D. Fremont, "Automated retraining of cyber-physical systems," 2022.

[45] AgileML Project, "Agileml: Agile machine learning framework," https://agileml.com/, 2024, accessed: 2025-04-07.

[46] E. Ostrom, *Governing the Commons: The Evolution of Institutions for Collective Action*, ser. Political Economy of Institutions and Decisions. Cambridge University Press, 1990.

Appendix A

Automated Imaging System for Microplastics Sample Classification Using Convolutional Neural Network

Project Deliverables

Deliverable Number	Details	Due Date
D1	XY Platform	September 12, 2025
D2	Revised Software Application	September 19, 2025
D3	Retrainable ML Model	September 30, 2025
D4	XY Platform – Software Integration	October 10, 2025
D5	Software – Retrainable ML Model Integration	October 13, 2025
D6	First Functional Prototype on different microscopes	October 17, 2025
D7	Initial User Feedback	October 24, 2025
D8	Second Functional Prototype with user feedback adjustments	November 17, 2025
D9	Second User Feedback	November 21, 2025
D10	Final Functional Prototype	December 2, 2025

Appendix B

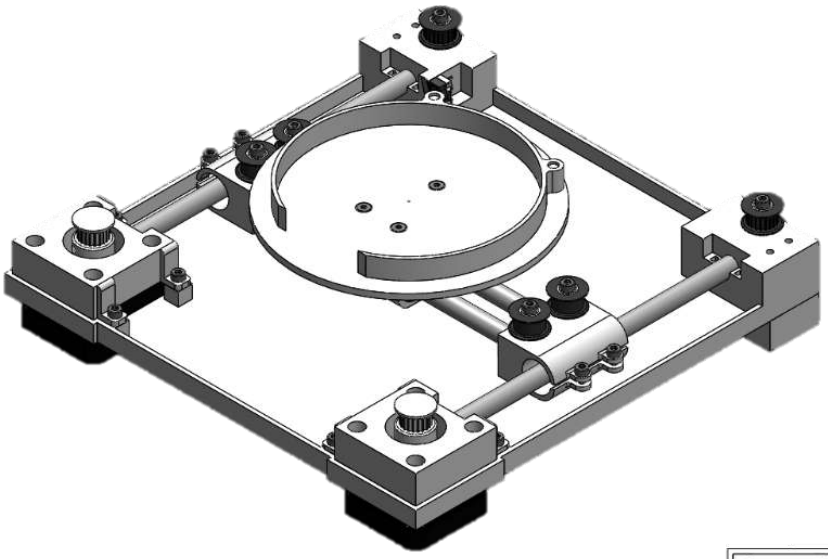
Automated Imaging System for Microplastics Sample Classification Using Convolutional Neural Network

Project Cost

Budget for Materials			
Item Description	**Quantity**	**Cost/Unit (PHP)**	**Subtotal (PHP)**
H1600 Digital Camera	1	4,699.00	4,699.00
3D Printed Components	1	2,000.00	2,000.00
NEMA 17 Stepper Motors	2	257.00	514.00
Arduino Uno	1	138.00	138.00
CNC Shield v3	1	129.00	129.00
DRV8825 Motor Driver	2	44.00	88.00
Limit Switch	2	5.00	10.00
Plug-in Power Supply	1	50.00	50.00
HDMI Cable	1	29.00	29.00
HDMI Capture Card	1	154.00	154.00
8mm Linear Rod (.582 meter)	1	257.00	257.00
LM8UU Linear Bearing	4	55.00	220.00
Gt2 Timing belt (2 meter)	1	220.00	220.00
20T Drive Pulley	2	56.00	112.00
20T Idler Pulley	4	64.00	256.00
M3 screw	29	2.00	58.00
		Total	**8,934.00**

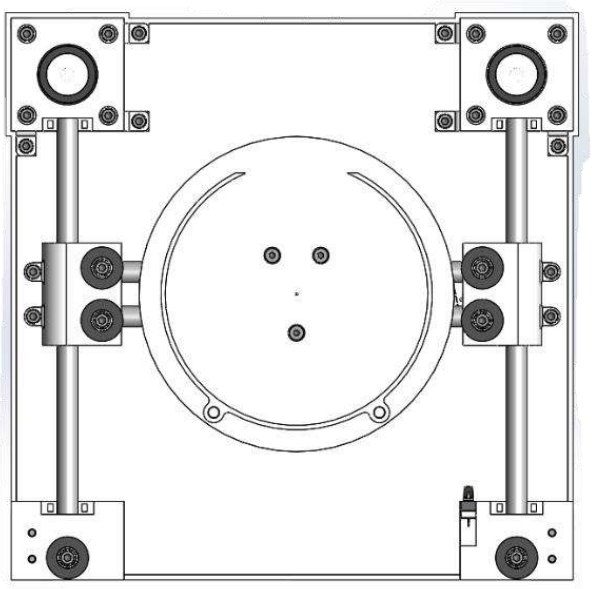
Appendix C

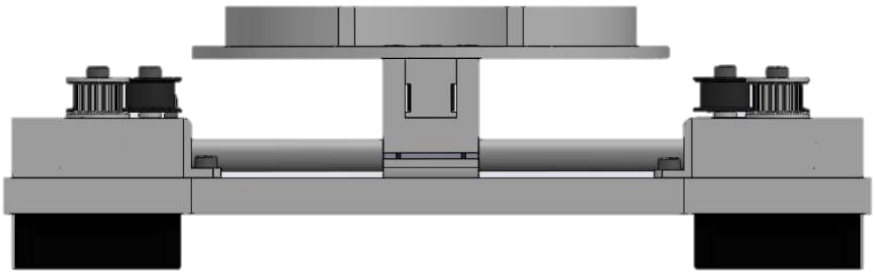
XY Platform



Isometric View

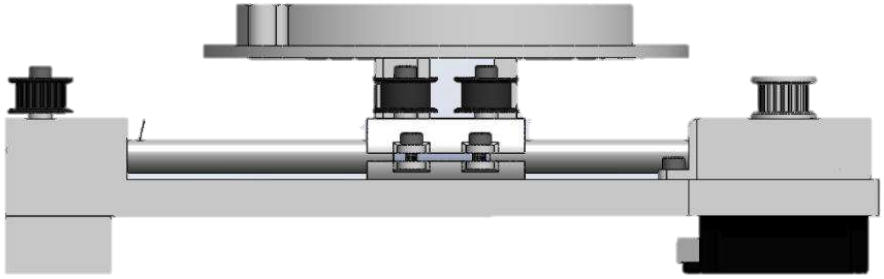
Top View

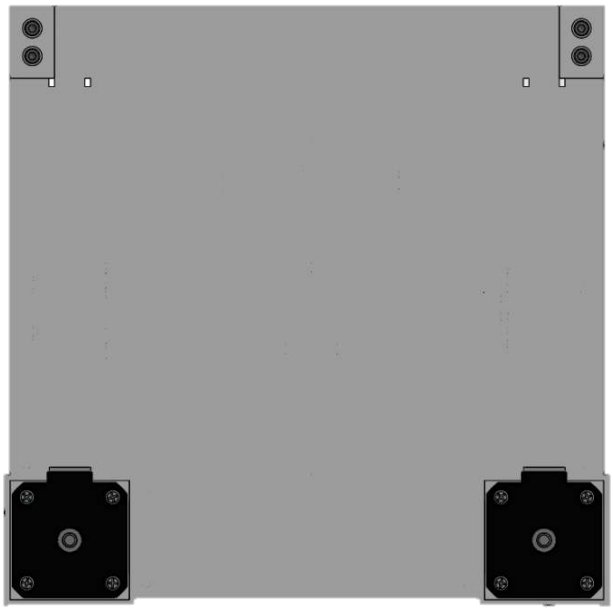




Front View

Side View





Bottom View

STUDENT-ADVISER THESIS COUNSELING LOGBOOK

Department of Computer Engineering
School of Engineering

Research Project
1nd, AY 2025 – 2026

Name of Adviser: Dr. Luis Gerardo S. Cañete
Name of Assistant Adviser: Dr. Philip Virgil B. Astillo

Name of Students:
Piolo Pascual E. Besinga
Ivor Louisetyne Canque
Lysander S. Uy

DATE	*PLACE / MEDIUM*	*TIME START/END*	*TOPIC DISCUSSED*	*ADVISER / ASST. ADVISER SIGNATURE*
June 20, 2025	PCB Laboratory	1:30 PM – 4:30 PM	- Research Updates	
June 27, 2025	PCB Laboratory	1:30 PM – 4:30 PM	- Research Updates	
July 04, 2025	PCB Laboratory	1:30 PM – 4:30 PM	- Research Updates	
July 15, 2025	PCB Laboratory	1:30 PM – 4:30 PM	- Research Updates	
July 24, 2025	PCB Laboratory	1:30 PM – 4:30 PM	- Research Updates	
July 31, 2025	PCB Laboratory	1:30 PM – 4:30 PM	- Research Updates	
August 08, 2025	PCB Laboratory	1:30 PM – 4:30 PM	- Research Updates	
August 26, 2025	PCB Laboratory	1:30 PM – 4:30 PM	- Research Updates - Consulted on Software Optimization	
September 04, 2025	PCB Laboratory	1:30 PM – 4:30 PM	- Research Updates - Consulted on the XY Platform - Consulted on Retraining Data	
September 11, 2025	PCB Laboratory	1:30 PM – 4:30 PM	- Research Updates - Consulted on the Belt Lock, and Limit Switches placement - Consulted on Benchmarking - Consulted on the New Hyperparameters	
September 18, 2025	PCB Laboratory	1:30 PM – 4:30 PM	- Research Updates	
September 25, 2025	PCB Laboratory	1:30 PM – 4:30 PM	- Research Updates - Consulted on Data Mining, Anchor Logic - Reviewing Project Deliverables	
October 04, 2025	PCB Laboratory	1:30 PM – 4:30 PM	- Research Updates	

			- Consulted on the Belt Tensioning - Benchmarking	
October 09, 2025	PCB Laboratory	1:30 PM – 4:30 PM	- Research Updates	
October 23, 2025	PCB Laboratory	1:30 PM – 4:30 PM	- Research Updates - Consulted on software-based tensioning and CNC Shield casing - Consulted on the first prototype	
October 30, 2025	PCB Laboratory	1:30 PM – 4:30 PM	- Research Updates - Consulted on the AnnotationWindow	
November 24, 2025	PCB Laboratory	1:30 PM – 4:30 PM	- Research Updates - Consulted on the New Look of the AnnotationWindow - Consulted on the benchmarking, new metrics, and retraining results	
December 03, 2025	PCB Laboratory	1:30 PM – 4:30 PM	- Reviewing Final Manuscript	

Automated Imaging System for Microplastics Sample Classification Using Convolutional Neural Network

Project Specifications

Main Project Function:

- Automate the movement of petri dishes using a custom X-Y platform to search for microplastic particles in samples.
- Utilize Artificial Intelligence (Convolutional Neural Networks) to automatically identify and classify microplastics found in samples.
- Provide a user interface for researchers to capture images, view identification results, and generate tabulated statistics.
- Streamline the microplastic research process to reduce physical strain and human error associated with manual determination.

Hardware

- Digital Microscope with built-in camera (e.g., Yizhan Scope-2) for high-resolution observation.
- Arduino Uno Microcontroller for controlling the platform logic.
- CNC Shield for Arduino to drive the stepper motors.
- NEMA 17 Stepper Motors (Quantity: 2) for X and Y axis movement.
- Custom 3D Printed X-Y Platform (PLA material) including brackets, sliders, and pulleys.
- Hardened Steel Linear Guides (151mm and 140mm) and LM8UU Bearings.
- GT2 Timing Belt and Pulleys for motion transmission.

Software

- Python for the server-side controller and machine learning model implementation.
- PyQt for the User Interface (View) following an MVC pattern.
- OpenCV for computer-vision tasks such as real-time video processing and image preprocessing.
- Convolutional Neural Network (CNN) architecture for microplastic sample identification.
- GRBL (G-code interpreter) firmware for controlling the X-Y table motion and camera focus.

Project Expectations

Target Users: Microplastic Researchers.

Performance: Minimum identification accuracy of 90% on the validation set after training.
Optimization: Reduce the time required for scanning a single sample compared to the manual process.

Safety Requirements: Include an "Emergency Stop" feature to immediately halt all GRBL movement and disconnect the firmware in case of emergency.

PROJECT TITLE: Automated Imaging System for Microplastics Sample Classification Using Convolutional Neural Network **DATE: 10/04/2025**

PROJECT MANAGER: Ivor Louisetyne Canque **GROUP: I**

WBS NUMBER	TASK TITLE	EXPECTED OUTPUT	JUNE				JULY				AUGUST				SEPTEMBER				OCTOBER				NOVEMBER				DECEMBER			
			WEEK 1	WEEK 2	WEEK 3	WEEK 4	WEEK 1	WEEK 2	WEEK 3	WEEK 4	WEEK 1	WEEK 2	WEEK 3	WEEK 4	WEEK 1	WEEK 2	WEEK 3	WEEK 4	WEEK 1	WEEK 2	WEEK 3	WEEK 4	WEEK 1	WEEK 2	WEEK 3	WEEK 4	WEEK 1	WEEK 2	WEEK 3	WEEK 4
1	**Project Definition and Planning**																													
1.1	Project Planning and Proposal	Project Plan																												
1.1.1	Definition of Project Objectives	RRL																												
1.2	Research	Project Outline																												
1.3	Revision of Design Documents	Design Documents																												
1.4	Development of Timeline and Project	Timeline and Project Deliverables																												
2	**Hardware Improvement**																													
2.1	Development of new X-Y Platform	New X-Y Platform 1st Prototype																												
2.2	X-Y Platform Adjustments from Compatibility Test	Adjusted New X-Y Platform 1st Prototype																												
2.3	X-Y Platform Adjustments from	New X-Y Platform 2nd Prototype																												
2.4	Final C.Scope.AI X-Y Platform Version 2	New X-Y Platform Final Prototype																												
3	**Software Improvement**																													
3.1	Implementing a version of PolyVision	PolyVision V1 without docker																												
3.2	PolyVision User Interface Revamp	PolyVision V2 1st Prototype																												
3.3	PolyVision revision based on Initial	PolyVision V2 2nd Prototype																												
3.4	Final PolyVision Version 2 Prototype	PolyVision V2 Final Prototype																												
4	**Machine Learning Improvement**																													
4.1	Machine Learning Retraining	CNN Model with Retraining 1st Prototype																												
4.2	1st Retraining Implementation	CNN Model with Retraining 2nd Prototype																												
4.3	Final C.Scope.AI Machine Learning	CNN Model with Retraining FinalPrototype																												
5	**System Integration**																													
5.1	Integration of New X-Y Platform and	X-Y Platform control in PolyVision																												
5.2	Integration Dockerless Polyvision	CNN Model with Retraning in PolyVision																												
5.3	First Complete Integration of	X-Y Platform Control & CNN Model in PolyVision (Prot 1)																												
5.4	Second Complete Integration of	X-Y Platform Control & CNN Model in PolyVision (Prot 2)																												
5.5	Final Complete System Integration of	Final X-Y Platform Control & CNN Model in PolyVision																												
6	**Feedback and Evaluation**																													
6.1	X-Y Platform Compatibility Test with Different	Assessed New X-Y Platform Compatibility																												
6.2	Initial User Feedback	Collected User Feedback of																												
6.3	Second User	Collected User Feedback of																												
6.4	Final User Feedback of Final System	Collected Final User Feedback of Final Prototype																												
7	**Documentation, Thesis Manuscript and Project Defense**																													
7.1	Initialization of Final Manuscript with	Started Working on Thesis Manuscript																												
7.2	Updated Final Manuscript with Data	Thesis Manuscript Updated with New Data																												
7.3	Complete Draft of Final Manuscript	Thesis Manuscript for Thesis Defense																												
7.4	Project Defense Presentation	PPT and Presentation for Thesis Defense																												
7.5	Final Thesis Defense	Thesis Defended																												
7.6	Manuscript Revisions	Panel guided paper revisions																												
8	**System Prototypes**																													
8.1	First System Prototype	C.Scope.AI Version 2 Prototype 1																												
8.2	Second System Prototype	C.Scope.AI Version 2 Prototype 2																												
8.4	Final Functional System Prototype	C.Scope.AI Version 2 Final Prototype for Deployment																												

Automated Imaging System for Microplastics Classification using Convolutional Neural Network

Binary			Multiclass		
Image	Live Feed	Difference	Image	Live Feed	Difference
99.8	99.74	0.06	73.46	85.82	-12.36
96.9	94.32	2.58	99.19	86.62	12.57
85.62	82.44	3.18	90.82	75.57	15.25
99.76	99.64	0.12	99.24	98.45	0.79
99.71	98.68	1.03	93.2	79.73	13.47
99.48	98.72	0.76	94.17	90.22	3.95
94.85	90.29	4.56	98.9	97.61	1.29
96.56	93.31	3.25	98.55	98.9	-0.35
99.79	99.62	0.17	97.26	82.92	14.34
99.23	98.57	0.66	84.06	70.63	13.43
92.87	97.68	-4.81	88.09	85.63	2.46
99.15	98.23	0.92	72.35	79.34	-6.99
97.43	98.76	-1.33	96.5	76.65	19.85
98.91	98.54	0.37	89.42	94.12	-4.7
96.64	76.76	19.88	94.08	70.81	23.27

Binary Model - 2.09 ± 5.38

Multiclass Model - 6.42 ± 10.43