VIETNAM NATIONAL UNIVERSITY OF HOCHIMINH CITY
THE INTERNATIONAL UNIVERSITY
SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

BUILDING AN AI MODEL TO SUPPORT STUDENTS FIND LOST THINGS

By

Mai Le Hung

ITITIU19125

A thesis submitted to the School of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
Bachelor of Computer Science

Ho Chi Minh City, Vietnam
2024

BUILDING AN AI MODEL TO SUPPORT STUDENTS FIND LOST THINGS

APPROVED BY: _____
Huynh Kha Tu, Ph.D.

_____

_____

_____

_____

THESIS COMMITTEE

ACKNOWLEGMENTS

I might want to offer my earnest thanks to Dr. Huynh Kha Tu, my boss, for his extraordinary direction and ardent help throughout the improvement of my postulation. I might likewise want to communicate my genuine thanks to Dr. Nguyen Quang Phu, my regarded specialist, whose experiences and useful criticism altogether upgraded the nature of the review. This. The whole staff at the International University School was additionally exceptionally praised for cultivating a rousing learning climate.

I might want to thank my loved ones for their unfaltering help and support; Their confidence in me is a spurring factor. Extraordinary acknowledgment is given to the members and supporters who assumed a critical part in the outcome of this venture. Every one of you lastingly affects this learning venture and for that, I am genuinely appreciative.

Table of Content

Tables of Figures

List of Tables

CHAPTER 1: INTRODUCTION

1.1 Background

In educational settings, students often face challenges related to misplacing personal items, such as textbooks, notebooks, or personal devices, disrupting their daily routines. The act of searching for lost items consumes considerable time and effort, affecting both academic and personal activities. Beyond inconvenience, this common occurrence detrimentally influences students' focus, productivity, and overall educational experience.

Acknowledging the widespread nature and impact of this problem, the current research aims to tackle the persistent issue of lost items in educational environments. The motivation behind this endeavor arises from the understanding that improving the efficiency of item retrieval can positively impact not only students' academic performance but also their overall well-being.

To address these challenges, the project delves into the application of advanced Artificial Intelligence (AI) techniques, specifically focusing on the YOLOv8 (You Only Look Once) model. This state-of-the-art object detection algorithm has the potential to transform the process of locating misplaced items by offering a mobile and accurate solution. Just anywhere, students can upload a photo of the object they are looking for and AI will determine for that student whether the object is in the student affairs room or not.

As educational institutions increasingly adopt technology to enhance various operations, incorporating an AI-driven system for item detection provides a timely and innovative approach to addressing a persistent and universal problem. This chapter sets the stage by introducing the contextual landscape, highlighting the need for a sophisticated yet practical solution to the issue of lost items among students, paving the way for the subsequent exploration of the proposed methodology and its implications.

1.2 Problem Statement

In spite of the strides made in educational technology, the ongoing challenge of students misplacing personal items in academic settings remains a significant obstacle. The prevalence of lost belongings disrupts students' daily routines, affecting both their academic performance and overall satisfaction. Conventional methods of item retrieval often prove to be time-consuming and inefficient, leading to frustration among both students and faculty. Additionally, contemporary students tend to be passive and hesitant to engage with school departments directly, often resorting to posting inquiries in online forums such as Facebook rather than seeking assistance through official channels.

Furthermore, every year the number of lost items brought to the Student Office is very large. Over the years, the number will be very huge, so directly finding objects lost many years ago will become difficult and time-consuming.

The absence of a systematic and technologically driven solution exacerbates this issue. Current approaches heavily rely on manual efforts and ad-hoc procedures, contributing to prolonged search times and heightened stress levels for students. As educational institutions continue to progress, there is an increasing demand for innovative solutions that leverage the capabilities of Artificial Intelligence to address practical challenges like the retrieval of lost items.

This project acknowledges the pressing need to implement an advanced system that integrates computer vision with AI capabilities, specifically utilizing the YOLOv8 model. The goal is to develop an intelligent and automated tool that not only accurately detects lost items but also expedites the process of disseminating information to students. By gaining a comprehensive understanding of the challenges associated with lost items in educational settings, this project aims to present a practical solution that aligns with the evolving landscape of modern educational institutions.

The creation of such a system represents a proactive initiative toward cultivating a conducive learning environment, minimizing unnecessary disruptions caused by lost belongings, and elevating the overall student experience.

1.3 Scope and objectives

1.3.1 Scope

This project is dedicated to creating and deploying an Artificial Intelligence (AI) solution utilizing the YOLOv8 model to address the prevalent issue of misplaced items in educational settings. The objectives encompass the development and implementation of a system with specific methodologies, facilitating the easy uploading of data by student affairs staff for AI training. Subsequently, users will only need to upload an image of the lost item to obtain search results. The intention is to craft a user-friendly interface that enables students and faculty to effortlessly access information and recover lost possessions.

By employing the YOLOv8 model, the project ensures a streamlined search process that accommodates the dynamic and diverse array of items found in educational environments. Additionally, the project scope includes considerations for integrating the system into the existing school infrastructure, promoting adaptability and accessibility.

1.3.2 Objectives

The main goal of this project is to create an application that can help students find their lost items without having to go directly to school. The first thing is to create an AI capable of recognizing objects. Furthermore, it must recognize and distinguish objects that have almost the same shape and color. In addition, it is necessary to improve model performance to save data and training time as well as item detection performance and accuracy. I will use the YOLOv8 model for this project and also use Annaconda as a software to support creating virtual environments for AI. There are also annotation support tools like LabelImg and libraries like PyTorch.

Next, I will create a UI linked to the trained AI and database. This will create convenience for all users as they do not need to understand computer algorithms to still use our application. This UI will be designed to be simple but clear and full of features so users can use it easily. This requires the UI to be able to receive the image file the user uploads and then launch the environment and AI to detect the item. Finally, it will return to the user an image of the item's detection results. Furthermore, the UI must continuously have notifications throughout the interaction with the user so that the user clearly understands the process and notes when using the software. To implement this UI, I decided on the PyQt5 library with many very good support features.

1.4 Assumption and Solution

Assumptions:

Infrastructure:

The system must be able to run on commonly available hardware today. Needs to be compatible with the school's and students' infrastructure.

Standardized Object Types:

The system assumes a degree of standardization in the types of objects typically found in educational settings. Common objects such as bags, books, and electronic devices are considered within the training dataset.

Student Cooperation:

The success of the system relies on the cooperation of students in reporting lost items. It is assumed that users will actively engage with the interface to input information about their lost belongings.

Solution:

Optimization with Existing Infrastructure:

The system needs to optimize input data to train AI. At the same time, split data into many separate objects to avoid problems with hardware limitations. Ensuring the system can run under current common hardware conditions.

Dynamic Object Recognition:

The YOLOv8 model, known for its versatility in detecting various objects, will be employed to address the assumption of standardized object types. The model's capability to adapt to diverse objects ensures flexibility in recognizing a wide array of items.

User-Friendly Reporting Interface:

To encourage student cooperation, a user-friendly interface will be developed, allowing students to easily find lost items. Clear and detailed instructions will be implemented to enhance user engagement.

1.5 Structure of thesis

This proposal is organized into the accompanying segments, each adding to a complete comprehension of the venture:

Chapter 1: Introduction

Chapter 1 Introduction is an outline of the whole of the postulation. It will dive into the specific situation, objectives, and direction of the undertaking. Moreover, it will cover the difficulties faced by the undertaking and the related arrangements that were executed. Finally, chapter 1 will introduce an outline of the construction of the proposition.

Chapter 2: Literature Review/Related Work

Chapter 2 will present the unique circumstances and content of certain ventures and examinations connected with my proposal. From that point, it provides peruses with a goal perspective on the venture's substance contrasted with other comparable activities.

Chapter 3: Methodology

This part subtlety the procedure utilized in the venture, including viewpoints, for example, information assortment, information arrangement, YOLOv8 model preparation, and generally speaking framework plan.

Chapter 4: Implementation and Results

This part portrays the execution of the artificial intelligence arrangement and presents the outcomes acquired through the discovery of lost objects. It incorporates conversations about execution measurements.

Chapter 5: Discussion and Evaluation

This chapter completely examines the outcomes, assesses the viability of the created framework, and investigates any difficulties experienced during execution.

Chapter 6: Conclusion and Future Work

This segment sums up the fundamental discoveries of the review, makes inferences, and examines possible future roads to improve and expand the proposed arrangement.

CHAPTER 2: LITURATURE REVIEW

The literature review in this chapter seeks to offer a thorough overview of pertinent research and related efforts in the realms of object detection, artificial intelligence (AI) applications in education, and solutions addressing the recovery of lost items. This segment serves as the cornerstone for comprehending the current knowledge landscape, shaping the context and contributions of the present project.

The exploration commences with an investigation into object detection technologies employed in educational environments. Recognizable algorithms and models applied in analogous contexts are pinpointed to establish a foundational understanding of the state-of-the-art in object detection. Subsequently, the literature review delves into the broader domain of AI applications in education, exploring how AI has been utilized to confront various challenges within educational settings.

Furthermore, the review scrutinizes existing solutions and technologies devised to meet the challenge of lost item retrieval. This involves an analysis of successful implementations and a critical examination of limitations observed in prior approaches. The incorporation of AI solutions into educational infrastructure is examined, taking into account associated challenges, benefits, and ethical considerations.

Moreover, the literature review extends to the realm of user interaction and interface design in systems akin to the proposed solution. Best practices and considerations for creating user-friendly interfaces are identified based on the examined literature. The section concludes with an exploration of challenges and limitations recognized in previous work, setting the stage for the synthesis of these findings in subsequent sections.

Through this literature review, the objective is to contextualize the current research within the existing body of knowledge, pinpoint gaps and opportunities for innovation, and establish the foundation for the subsequent chapters, particularly the methodology that builds upon the insights garnered from this comprehensive review.

OBJECT DETECTION WITH YOLOV8:

The YOLOv8 model, introduced by Redmon and Farhadi in 2018, represents a significant leap in real-time object detection capabilities. Notably, its successive versions, such as YOLOv4 (Wang & Zhang, 2020), showcase a commitment to optimizing both speed and accuracy. YOLOv8's modular architecture and efficient processing make it an appealing choice for applications requiring rapid identification and localization of objects. The model's adaptability and consistent updates underline its relevance in the dynamic field of computer vision, forming the cornerstone of the proposed AI solution for lost item retrieval [1].

YOLOV4: OPTIMIZING SPEED AND ACCURACY IN OBJECT DETECTION:

The YOLOv4 model, presented by Wang and Zhang in 2020 [2], marks a significant advancement in object detection, achieving optimal speed and accuracy. Introduced as an evolution of the YOLOv3 architecture, YOLOv4 integrates improvements in both model

architecture and training techniques. This preprint on arXiv (arXiv:2004.10934) provides essential insights into the model's design choices and optimization strategies, serving as a valuable reference for enhancing the YOLOv8 model's performance in real-time lost item detection [2].

DEEP LEARNING FRAMEWORKS:

PyTorch, a pivotal component in this project's framework, stands out as a dynamic deep learning library known for its flexibility and expansive community support. Developed by Facebook, PyTorch facilitates seamless neural network design, training, and implementation. Its intuitive and dynamic computation graph, in contrast to static alternatives, simplifies model development. The rich set of pre-built functions and modules accelerates the implementation of complex architectures like YOLOv8 [3][4]. PyTorch's extensive documentation provides valuable insights into leveraging its features effectively. The platform's popularity in the AI community ensures a wealth of resources, tutorials, and collaborative expertise, making it an indispensable tool in the development of the proposed AI model for lost item detection and retrieval.

GRAPHICAL USER INTERFACE (GUI) DEVELOPMENT:

PyQt5, an essential component in this project, empowers the development of a user-friendly Graphical User Interface (GUI). This Python library streamlines the creation of desktop applications, offering a seamless blend of functionality and aesthetics. Its comprehensive documentation [5] serves as a guide for integrating intuitive interfaces, ensuring an engaging user experience. The incorporation of PyQt5 enables efficient interaction with the AI model, enhancing accessibility for students seeking lost items through a visually appealing and responsive interface.

PYTHON AND ANACONDA ECOSYSTEM:

Python, at the core of this project, aligns with the prevailing trend in AI development. Its versatility, extensive libraries, and community support make it an ideal choice for implementing complex machine learning solutions. Anaconda, a powerful distribution of Python, enhances project management by simplifying package installations and handling dependencies [6]. This ecosystem provides a robust environment, ensuring seamless integration of tools and libraries necessary for building and deploying the proposed AI model for lost item detection, offering a cohesive and efficient development experience.

DIGITAL IMAGE PROCESSING:

Gonzalez, Woods, and Eddins' work on digital image processing using MATLAB (2009) lays foundational principles that extend to the proposed AI model. Although the project utilizes Python, the fundamental concepts of image processing elucidated in their book serve as a valuable guide. Understanding image enhancement, segmentation, and feature extraction from their comprehensive perspective enriches the project's approach to preprocessing data for YOLOv8. This reference becomes an essential resource for adapting and implementing digital

image processing techniques within the Python environment, contributing to the model's accuracy in detecting and localizing misplaced items [7].

LARGE-SCALE VISUAL RECOGNITION CHALLENGES:

The ImageNet Large Scale Visual Recognition Challenge (ILSVRC), spearheaded by Russakovsky et al. in 2015, has been pivotal in advancing object detection research. ILSVRC's influence transcends the specific challenges posed to participating models, serving as a benchmark for evaluating the efficacy of object detection algorithms. Pre-trained models on ImageNet, such as those derived from ILSVRC, often serve as a foundational starting point for fine-tuning models like YOLOv8 [8]. The extensive and diverse ImageNet dataset has played a crucial role in shaping the robustness of object detection models, ensuring their ability to generalize across varied scenarios. Insights gained from ILSVRC guide the adaptation of the proposed AI model to real-world conditions, enhancing its capacity to recognize and locate lost items with accuracy and efficiency. Leveraging the lessons learned from large-scale challenges contributes to the continual evolution of the YOLOv8 model within the broader landscape of visual recognition [8][9].

SINGLE SHOT MULTIBOX DETECTOR (SSD) AND MASK R-CNN:

Single Shot MultiBox Detector (SSD), introduced by Liu et al. in 2016, and Mask R-CNN, proposed by He et al. in 2017, represent notable alternatives to object detection, each with distinctive strengths. SSD excels in efficiency, providing real-time object detection by predicting multiple bounding boxes and class scores in a single forward pass. Its single-shot approach significantly simplifies the detection pipeline, making it suitable for applications requiring low latency and high throughput [10].

Mask R-CNN, on the other hand, extends the capabilities of traditional object detectors by incorporating an additional segmentation branch. This enables precise instance-level segmentation, producing pixel-wise masks for identified objects. While Mask R-CNN excels in detailed scene understanding, it typically operates at a slower speed compared to SSD [11].

Insights from both SSD and Mask R-CNN contribute to a holistic understanding of object detection techniques. The trade-offs between speed and segmentation accuracy inform the decision to employ YOLOv8 in the current project, striking a balance suitable for real-time lost item detection and localization [10][11]. These models' methodologies inspire considerations in fine-tuning and optimizing YOLOv8, ensuring the proposed AI solution attains a harmonious equilibrium between speed, accuracy, and segmentation capabilities.

CONVOLUTIONAL NEURAL NETWORKS IN LARGE-SCALE VIDEO CLASSIFICATION:

In their seminal work on large-scale video classification, Karpathy et al. (2014) [12] explore the application of convolutional neural networks (CNNs) in processing extensive video datasets. Presented at the IEEE conference on Computer Vision and Pattern Recognition, their research provides crucial insights into the capabilities of CNNs for video analysis. This study becomes relevant in the context of the proposed AI model, informing potential adaptations for

comprehensive scene understanding and object detection, particularly beneficial in scenarios where lost items may be embedded within dynamic visual environments [12].

DEEP LEARNING ARCHITECTURES:

Deep learning architectures play a pivotal role in shaping the performance of computer vision models. VGG16, introduced by Simonyan and Zisserman in 2014, presents a deep convolutional network with a standardized structure, emphasizing the importance of depth in neural network architectures. Its simple yet effective design has influenced subsequent models, contributing to the understanding of feature hierarchies and receptive fields in image processing [13].

The Inception architecture, particularly InceptionV3, introduced by Szegedy et al. in 2016, explores the benefits of parallelized convolutions at different scales. This innovation mitigates the trade-off between computational cost and performance, fostering the development of efficient and accurate models for image classification tasks [13] [14].

While YOLOv8 remains the primary model for object detection in the current project, insights from VGG16 and Inception architectures inform potential refinements. Understanding feature extraction and hierarchical representation aids in optimizing YOLOv8 for improved lost item detection. The adaptability and cross-pollination of ideas across deep learning architectures contribute to a nuanced approach in enhancing the proposed AI model's capabilities [13] [14].

EXTRACTING INSIGHTS:

Goodfellow, Bengio, and Courville's comprehensive work, "Deep Learning" (2016) [15], stands as a foundational resource in the field. This seminal book delves into the theoretical underpinnings and practical applications of deep learning, elucidating core concepts such as neural network architectures, optimization algorithms, and training methodologies. The thorough exploration of these fundamentals becomes instrumental in the development of the proposed AI model. By referencing this authoritative text, the thesis draws upon the collective expertise of the authors to inform decisions regarding model design, training strategies, and potential enhancements. The integration of insights from "Deep Learning" ensures a robust theoretical framework, aligning the AI model with established principles and best practices in deep learning, thereby contributing to the model's efficacy in lost item detection [15].

OPTIMIZATION ALGORITHMS:

The choice of optimization algorithm profoundly influences the training efficiency and convergence of deep neural networks. Adam, introduced by Kingma and Ba in 2014, stands out for its adaptive learning rates and momentum, addressing challenges associated with traditional stochastic gradient descent methods. In the context of the proposed AI model using YOLOv8, the application of the Adam optimizer [16] is crucial. Its adaptive nature ensures efficient weight updates during training, facilitating the convergence of the model for accurate and rapid lost item detection. The optimization algorithm plays a vital role in enhancing the overall performance and training dynamics of the AI model.

DATASETS FOR OBJECT DETECTION:

Datasets for object detection serve as critical benchmarks for training and evaluating models. The PASCAL Visual Object Classes (VOC) Challenge, established by Everingham et al. in 2010, features diverse object categories and annotated images, providing a foundational dataset for object detection tasks. Its role extends beyond benchmarking, as it aids in fine-tuning models like YOLOv8 for improved accuracy and generalization [17].

The Common Objects in Context (COCO) dataset, introduced by Lin et al. in 2014, further enriches the understanding of object detection algorithms. COCO's large-scale collection includes complex scenes with multiple object instances and intricate annotations, fostering the development of models capable of handling real-world scenarios [9].

In the context of the proposed AI model, leveraging insights from both VOC and COCO datasets ensures a robust and versatile approach to lost item detection. The diverse range of object categories and scene complexities encapsulated in these datasets contributes to the adaptability and effectiveness of the YOLOv8 model, enhancing its capacity to recognize and locate misplaced items accurately [9][17].

INNOVATIONS IN CONVOLUTIONAL NETWORKS:

Recent innovations in convolutional networks have significantly shaped the landscape of computer vision, influencing the proposed AI model's architecture. DenseNet, introduced by Huang et al. in 2017, emphasizes dense connections between layers, promoting feature reuse and information flow throughout the network. This architecture enhances feature propagation and mitigates the vanishing gradient problem, potentially inspiring improvements in the YOLOv8 model's ability to capture intricate spatial relationships in images [18].

Squeeze-and-Excitation Networks (SENet), proposed by Hu et al. in 2018, introduce attention mechanisms within each channel of a convolutional layer. This innovative approach allows the model to dynamically adjust its focus on different features, improving the model's discriminative power. While YOLOv8 focuses on efficiency, insights from SENet can guide the exploration of attention mechanisms to refine object detection and localization capabilities [19].

The integration of these innovations aligns with the continual evolution of deep learning architectures, particularly in addressing challenges related to information flow and feature extraction. While YOLOv8 remains the primary model for object detection, the adaptability and cross-pollination of ideas from DenseNet and SENet contribute to a nuanced approach in optimizing the proposed AI model. The exploration of attention mechanisms and dense connections enhances the YOLOv8 model's ability to capture intricate features, potentially improving its accuracy and performance in locating lost items within diverse environments [18][19].

COMMUNITY INVOLVEMENT AND PROBLEM RESOLUTION:

Active participation in the YOLOv8 GitHub repository [20] provides a unique avenue for community involvement and collaborative problem resolution. Engaging with the repository

allows access to ongoing discussions, insights from developers, and solutions to potential challenges encountered during the project. The collaborative nature of the community fosters a dynamic environment where researchers and practitioners can collectively contribute to refining and advancing the YOLOv8 model, ensuring a robust and well-supported foundation for the proposed AI solution for lost item detection.

CHAPTER 3: METHODOLOGY

This chapter delineates the methodical approach employed for crafting an AI solution dedicated to the recovery of lost items in educational settings. Aligned with the research objectives, this approach prioritizes transparency and replicability. The essential components encompass study design, data collection, YOLOv8 model training, system implementation, assessment metrics, user interface design, and data analysis. Each facet contributes to the development of a pragmatic and user-friendly system. The significance of this methodology lies in its all-encompassing nature, addressing technical, ethical, and user-centric considerations to fulfill the research objectives. The ensuing sections offer in-depth insights into specific methodological nuances, furnishing a lucid roadmap for conducting the research.

3.1 Research Design

3.1.1 Overview

To address the problem of retrieving lost items in an educational setting, a research design was chosen to evaluate the effectiveness of the developed AI solution. This design of mine will go through the main parts of research first, then the general implementation part and finally the evaluation part. This design ensures that the project has a solid enough foundation of knowledge to optimize the implementation process. At the same time, there is a final evaluation step to evaluate the results and possibly outline future directions for the project.

3.1.2 Learn and research knowledge

Before starting a project, we must have a necessary amount of knowledge in the fields of machine learning, deep learning, UI, ... Only when we master this knowledge and master the necessary technologies can we project. Only then can this project be launched and implemented smoothly. Learning and analyzing to clearly understand the algorithm and then implementing it will minimize errors that arise throughout the project.

3.1.3 Design implementation plan

After acquiring the necessary knowledge and skills, this project needs a clear implementation plan so that the stages are carried out continuously and without confusion. A detailed plan from start to finish is extremely necessary for a project. As long as there is an accurate plan, the project will almost certainly be implemented smoothly.

3.1.4 Implementation and debugging

Next is the project implementation stage based on the proposed plan. However, throughout the implementation period, there will always be new updates or errors that were not present in the planning stage. Therefore, updating and fixing errors must always be done in parallel throughout the duration of the project.

3.1.5 Result evaluation

After the completion of the project, the re-evaluation is very necessary. This assessment will include accuracy, level of completion, UI, research results, etc. This will show an overview of the entire thesis and show the directions of post-thesis development if any.

3.1.6 Limitations of Research Design

Because my design for this project is done from my subjective perspective, there may be many different opinions about the framework and direction of the project. The development of technology is continuous, so the design may be more outdated than it is today.

3.2 YOLOv8 Architecture

3.2.1 Model Structure

YOLOv8 architecture is built with 3 main parts with separate and directly related functions: Backbone, Neck and Head.

Backbone:

First is the Backbone, the previous part of the model uses the CSP-Darknet53 structure. This part will be responsible for the first most parts of the model. It will process and extract information from the input data image. The backbone does this using a series of convolutional layers, extracting features such as edges, textures, and shapes. From there, the data will be classified into 5 separate layers in ascending order with different complexity. This hierarchical representation is due to the Cross-Stage Partial connections of the CSP-Darknet53 structure which allows for an associative and hierarchical representation of the input.

Neck:

Use SPPF and New CSP-PAN Structures. Neck is the part responsible for connecting the extracted features of the input image from the Backbone part with the Head part. This is an extremely important intermediate part that creates favorable conditions for information flow and allows the model to capture contextual relationships. SPPF allows synthesizing features at many different scales for models can look at data at many different resolutions. Path Aggregation is optimized by CSP-PAN Structures by combining features from different paths in the network. This enhances the model's understanding of the context in which objects appear.

Head:

The role of the Head is to generate predictions using features derived from both the Backbone and Neck components. Within the YOLO Head architecture, features undergo processing to anticipate the positioning of bounding boxes around objects within the image. Two primary parameters are involved: Class Prediction and Confidence Score. The Class Prediction estimates the likelihood of each identified object belonging to a specific class, thereby indicating

the object type. The confidence score signifies the model's level of certainty regarding the existence of an object, assisting in the exclusion of predictions with low confidence. Ultimately, the Head produces the final output, encompassing coordinates of bounding boxes, class probabilities, and confidence scores.

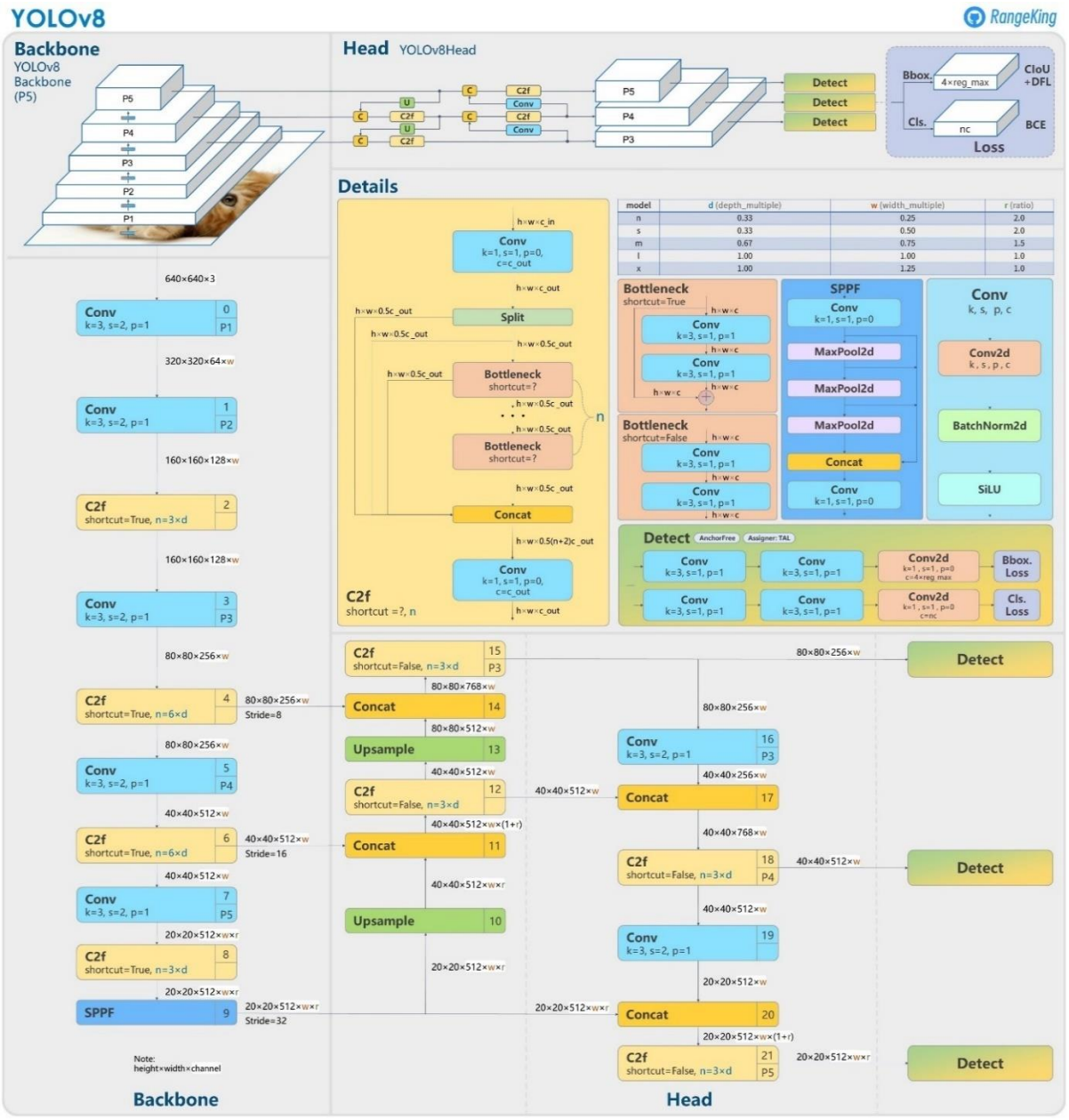

Figure 1 YOLOv8 Model Structure

3.2.2 Model Training

3.2.2.1 Data Augmentation

To improve, diversify and add random elements to the input data used to train the AI, the model has integrated features such as random flip, rotation, zoom and automatic brightness change, contrast. every time you start working out. This is essential, it will expand the ability of the trainer model to deliver broader results. Data augmentation plays an essential role in helping to learn from a diverse and highly accurate data set for a variety of input problems.

Figure 2 Random flip, rotation, zoom and automatic brightness change, contrast

3.2.2.2 Splitting the Dataset

To optimize and increase the accuracy of parameters and the ability to evaluate and predict the model, I have split the dataset into 2 separate parts: Training set and Validation set:

The training set constitutes a segment of the dataset utilized for instructing the YOLOv8 model. Throughout the training process, the model refines its predictive abilities and adjusts parameters by scrutinizing the patterns and relationships inherent in this subset. A more extensive training set typically empowers the model to grasp a broader range of diverse and representative features, thereby improving its capacity to extrapolate to novel, unseen data. Ensuring the training set is both sizable and varied is imperative to guarantee the model acquires robust features, enabling effective performance across a multitude of inputs.

The validation set is employed to assess the YOLOv8 model's performance during its training phase. Serving as an independent dataset not exposed to the model during its learning period, the validation set facilitates the computation of metrics such as accuracy, precision, recall, or custom evaluation metrics. These metrics offer insights into how effectively the model generalizes to unfamiliar, unseen data. By evaluating the model on the validation set, the risk of overfitting is mitigated—a circumstance where the model becomes overly tailored to the training data, resulting in suboptimal performance on new and unseen data.

3.2.2.3 Grid Division

The division of the image into a grid is a fundamental element of the YOLO algorithm, enabling it to effectively locate and categorize objects in an image. This involves organizing the input image into a regular grid of cells, with each cell tasked with predicting objects within its designated spatial area. This grid division is essential for the effectiveness of YOLO in object detection. The image is segmented into a grid (squares) based on the parameter *S*, which signifies the quantity of cells along each spatial dimension. The overall count of grid cells is computed as *S x S*. The *(x, y)* coordinates are expressed in relation to the dimensions of the grid cell, where *(0, 0)* denotes the top-left corner of the cell, and *(1, 1)* denotes the bottom-right corner. By segmenting the image into a grid and signing specific cells with the responsibility for predictions, YOLO achieves rapid and simultaneous object detection at various spatial positions. This methodology enhances YOLO's capacity for real-time processing, making it particularly suitable for applications where speed is crucial, such as in video analysis and autonomous systems.

3.2.2.4 Compute Losses

The algorithm for computing losses in YOLOv5 involves three key components: Classes Loss (BCE Loss), Objectness Loss (BCE Loss), and Location Loss (Complete IoU Loss).

Classes Loss (BCE Loss): This element evaluates the deviation in predicting class probabilities for each object and employs Binary Cross-Entropy Loss to measure the dissimilarity between predicted and actual class probabilities.

Objectness Loss (BCE Loss): This section quantifies the error in discerning the presence of an object within a specific grid cell. Binary Cross-Entropy Loss is utilized to assess the difference between predicted and true objectness scores.

Location Loss (Complete IoU Loss): The Complete IoU Loss gauges the inaccuracy in pinpointing the object within the grid cell. It calculates the Intersection over Union (IoU) between predicted and target bounding boxes, considering both their intersection and the surrounding

region. The Complete IoU Loss is then derived as the disparity between IoU and a term involving the enclosed area.

These three losses are amalgamated to create the overall loss by this function:

$$Loss = \lambda_1 L_{cls} + \lambda_2 L_{obj} + \lambda_3 L_{loc} \text{ (1)}$$

In that:

● L_{cls}: This term represents the classification loss. It measures the difference between predicted class probabilities and the true class labels. This part of the loss ensures that the model correctly identifies the class of objects in the image.

● L_{obj}: Objectness loss penalizes the model for failing to detect an object or generating a false positive. It incentivizes the model to predict accurately whether an object is present within a specified bounding box.

● L_{loc}: The localization loss measures the difference between predicted bounding box coordinates (e.g., x, y, width, and height) and the ground truth bounding box coordinates. This loss component ensures precise localization of objects in the image.

● $\lambda_1, \lambda_2, \lambda_3$: These are hyperparameters determining the importance of each loss component in the overall model loss. Typically configured during model training, they balance the impact of classification, objectness, and localization losses.

The “Loss” is employed in adjusting the model parameters during training via backpropagation. The objective is to minimize this collective loss, enhancing the model's capacity to precisely classify and localize objects within the YOLOv8 framework.

3.2.2.5 Balance Losses

The strategy of "Balance losses" entails assigning unique weights to objectness losses from three prediction layers (P3, P4, P5) in the YOLO architecture. The specified balance weights [4.0, 1.0, 0.4] dictate the relative importance of objectness losses from each layer in contributing to the overall loss. This method ensures that errors in predicting the presence or absence of objects at different scales exert distinct influences on the total loss. Specifically, a higher weight (4.0) is applied to the objectness losses from P3, signifying a more significant impact on the overall loss. In contrast, standard (1.0) and lower (0.4) weights are assigned to P4 and P5, respectively. This weighted integration seeks to achieve equilibrium in considering predictions across various scales, preventing excessive focus on a particular resolution, and fostering more effective learning across diverse spatial contexts. The balance loss of object is calculated by this function:

$$L_{obj} = 4.0 \cdot L_{obj}^{small} + 1.0 \cdot L_{obj}^{medium} + 0.4 \cdot L_{obj}^{large} \text{ (2)}$$

In that:

- L_{obj}^{small}: The objectness loss for small objects.
- L_{obj}^{medium}: The objectness loss for medium objects.
- L_{obj}^{large}: The objectness loss for large objects.

In this formulation, there's a proposal to assign distinct weights to the objectness loss concerning small, medium, and large objects. The weights, namely 4.0, 1.0, and 0.4, respectively, express the relative significance of the objectness loss for these varying size categories. The higher weight assigned to small objects implies a stronger penalty for detection errors in small objects compared to medium and large ones.

By adjusting these weights within the loss function, practitioners gain control over the influence of objectness loss across diverse object sizes. This weighting approach is commonly employed to address dataset imbalances or prioritize the detection of specific objects based on their size or significance in a given application.

3.2.2.6 Eliminate Grid Sensitivity

In YOLOv8, one significant change involves the use of anchor boxes for box prediction. Anchor boxes are predetermined bounding box shapes with specific widths and heights. Instead of directly predicting the raw coordinates of bounding boxes, YOLOv8 predicts adjustments (offsets) to these anchor boxes. In YOLOv2 and YOLOv3, the direct prediction of box coordinates was accomplished through the activation of the final layer:

$$b_x = \sigma(t_x) + c_x \text{ (3)}$$

$$b_y = \sigma(t_y) + c_y \text{ (4)}$$

$$b_w = p_w \cdot e^{t_w} \text{ (5)}$$

$$b_h = p_h \cdot e^{t_h} \text{ (6)}$$

In that:

- b_x, b_y: The center coordinates of the bounding box.
- b_w, b_h: The width and height of the bounding box.
- c_x, c_y: The coordinates of the grid cell.
- t_x, t_y: The coordinates of predicted offsets.
- t_w, t_h: The width and height of predicted offsets.

● σ: The sigmoid activation functions. Defined as:

$$\sigma(x) = \frac{1}{1+e^{-x}} \quad (7)$$

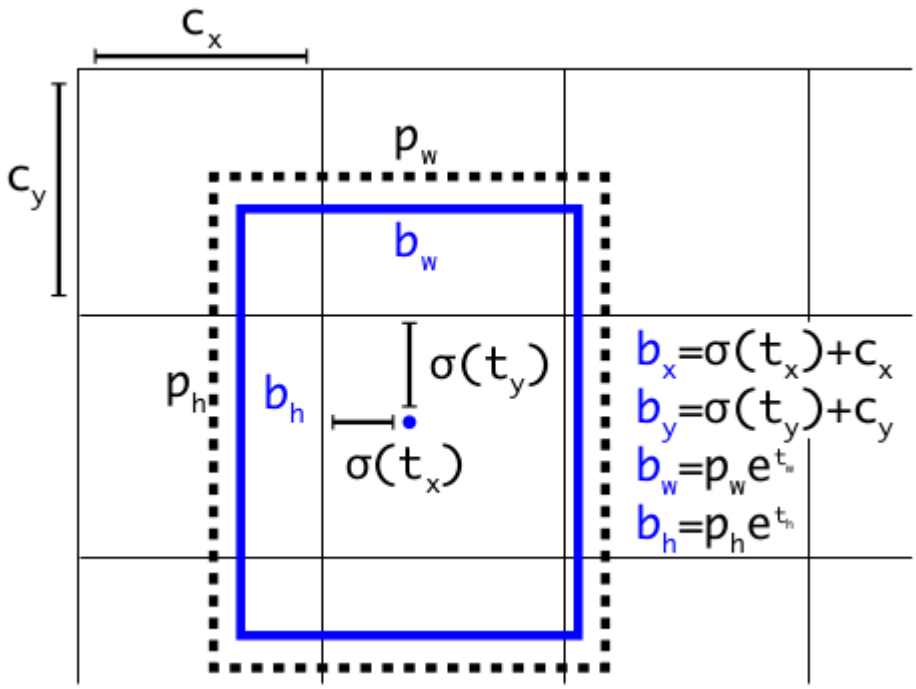

Figure 3 Eliminate grid sensitivity

Nevertheless, YOLOv5 has introduced an updated formula for predicting box coordinates, aiming to decrease grid sensitivity and prevent the model from predicting unbounded box dimensions. The modified equations for determining the predicted bounding box are outlined below:

$$b_x = (2 \cdot \sigma(t_x) - 0.5) + c_x \quad (8)$$

$$b_y = (2 \cdot \sigma(t_y) - 0.5) + c_y \quad (9)$$

$$b_w = p_w \cdot (2 \cdot \sigma(t_w))^2 \quad (10)$$

$$b_h = p_h \cdot (2 \cdot \sigma(t_h))^2 \quad (11)$$

In that:

● p_h, p_w: The width and height of the anchor box associated with the predicted bounding box.

Contrast the center point offset before and after scaling. The range of the center point offset is modified from (0, 1) to (-0.5, 1.5). Consequently, the offset can readily attain values of 0 or 1.

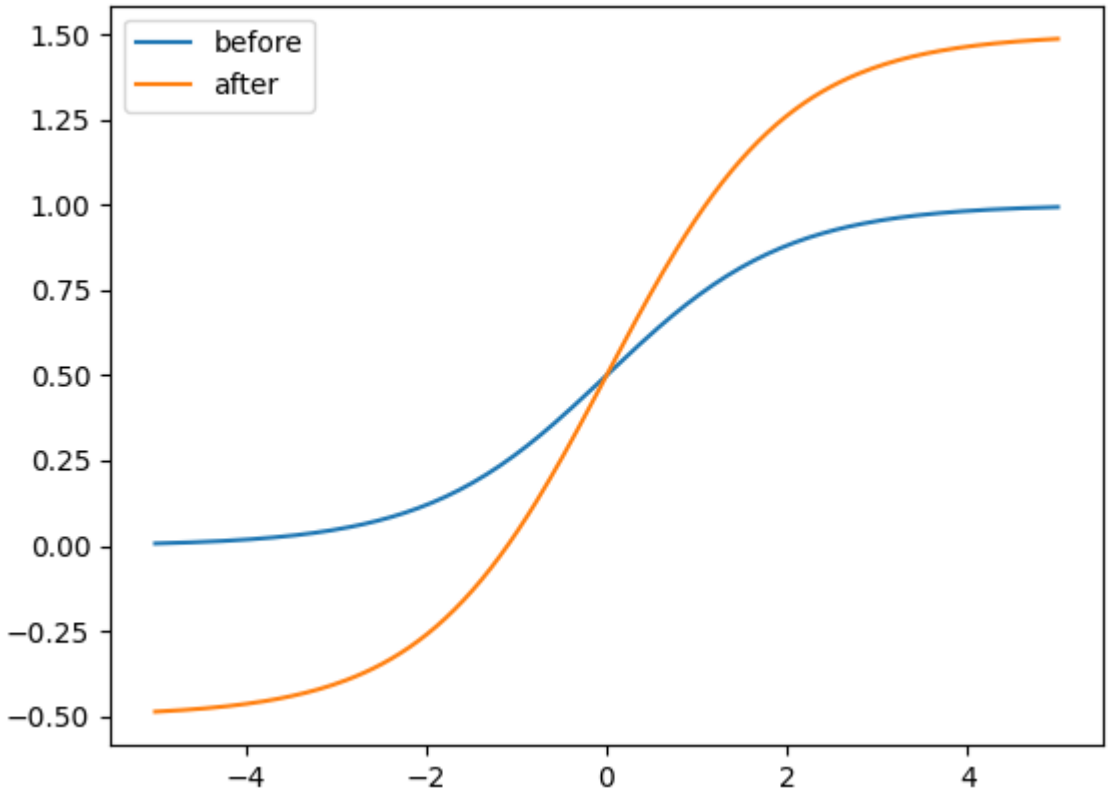

Figure 4 Compare the center point offset before and after scaling

Compare the scaling ratio of height and width concerning the anchor before and after modification. The initial YOLO/Darknet box equations present a notable issue. Width and Height are entirely unbounded since they are expressed as out=exp(in), posing a risk of runaway gradients, instabilities, NaN losses, and ultimately compromising the entire training process.

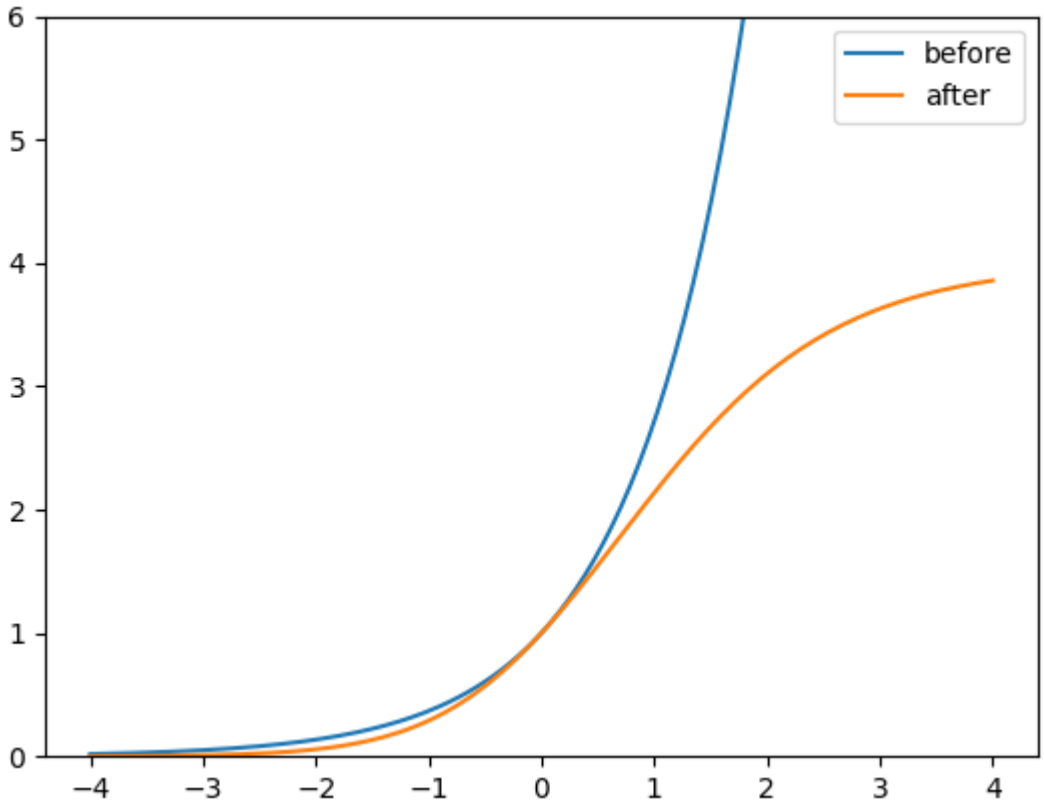

Figure 5 Comparison of the height and width scaling ratio (relative to anchor) before and after adjustment

3.2.2.7 Build Targets

The goal building process in YOLOv5 is important for training efficiency and model accuracy. It involves having the appropriate grid cells in the output map assigned with ground truth boxes then matching them to the appropriate anchor boxes. This process will be calculated according to the following steps:

First, calculate the ratio of the ground truth box dimensions and then is the dimensions of each anchor template:

$$r_w = w_{gt}/w_{at} \quad (12)$$

$$r_h = h_{gt}/h_{at} \quad (13)$$

$$r_w^{max} = \max(r_w, 1/r_w) \quad (14)$$

$$r_h^{max} = \max(r_h, 1/r_h) \quad (15)$$

$$r^{max} = \max(r_w^{max}, r_h^{max}) \quad (16)$$

$$r^{max} < anchor_t \ (17)$$

In that:

- r_w: Ratio of the ground truth bounding box width to the anchor box width.
- r_h: Ratio of the ground truth bounding box height to the anchor box height.
- w_{gt}, h_{gt}: Ground truth bounding box width and height.
- w_{at}, h_{at}: Anchor box width and height.
- r_w^{max}, r_h^{max}: The maximum width and height ratio.
- r^{max}: The overall maximum ratio.
- max (): This function will return the largest value among the provided inputs.
- $anchor_t$: The threshold or limit for the maximum allowed ratio.

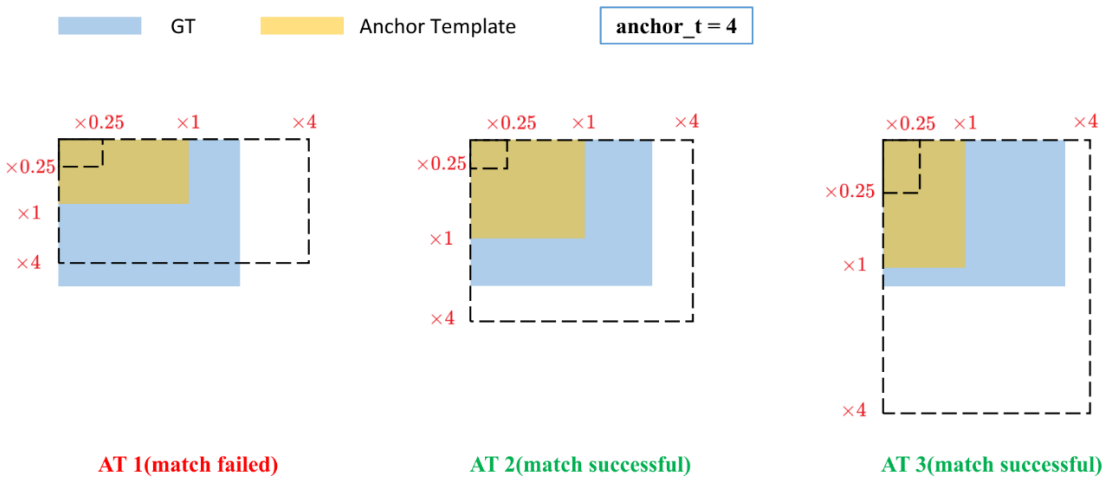

Figure 6 The ratio example calculations

Match the ground truth box with the associated anchor if the calculated ratio falls within the threshold:

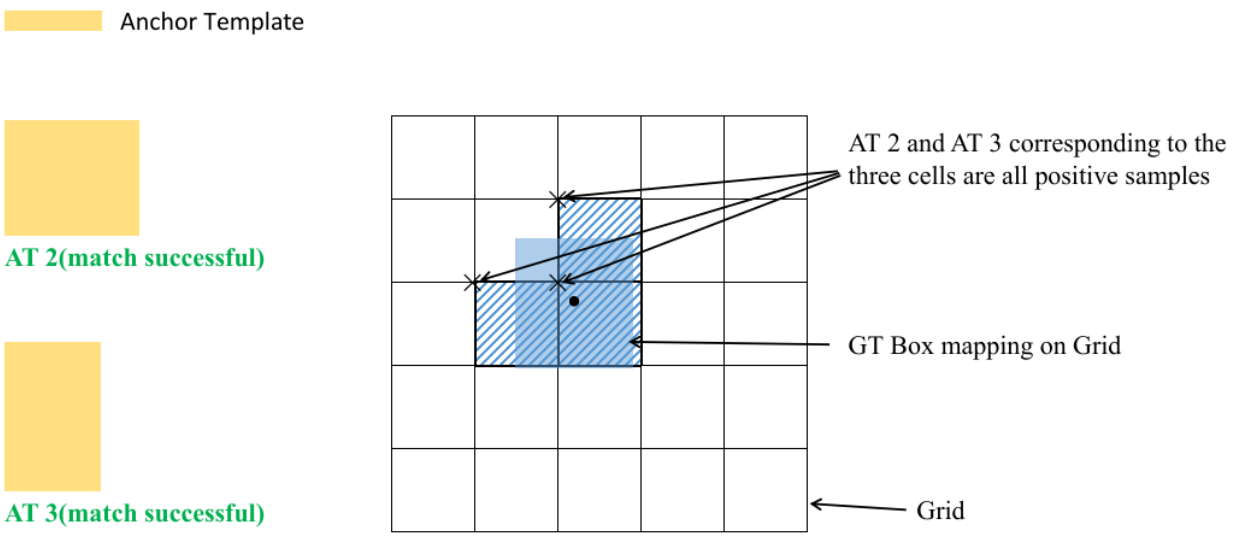

Figure 7 The ground truth box example matching

A ground truth box can be assigned multiple anchor points because the central offset is modified thereby assigning the appropriate anchor point to the appropriate cells:

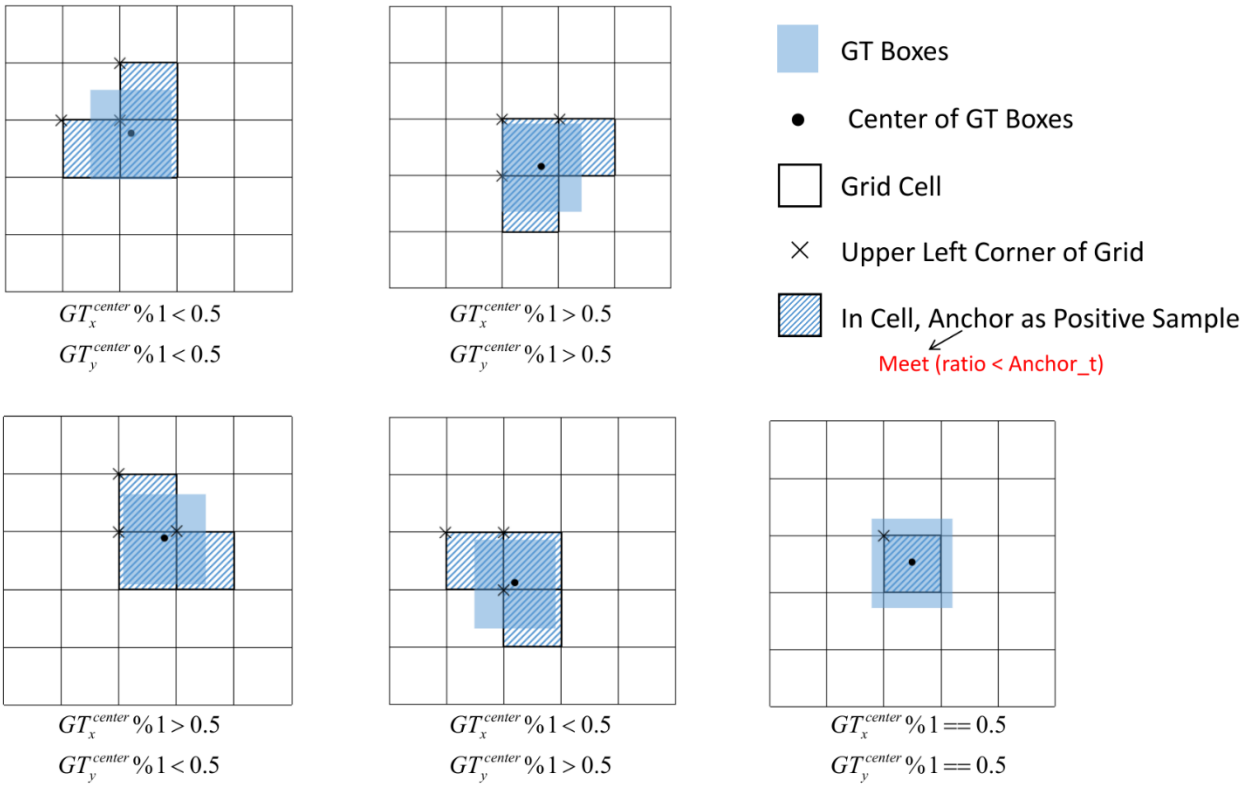

Figure 8 Ground truth box assign example

Thereby the build target process is guaranteed to be specified and matched accurately during the training process. Since then, the YOLO model has improved its accuracy in detecting objects.

3.3 Technologies

3.3.1 Python Programming Language

During the execution of this venture, Python will act as the main programming language. Famous for its effectiveness and undeniable level of usefulness, Python flaunts a grammar that isn't just basic yet in addition different, adaptable, clear, and easy to understand. First presented in 1991 by Guido van Rossum, Python immediately rose through the positions to all around the world become one of the most broadly utilized programming dialects. Its accentuation on clarity makes it particularly reasonable for the two novices and old pros the same. Besides, Python flaunts a broad library with endless elements and apparatuses, particularly outstanding in the field of Artificial Intelligence (AI). Its similarity with various stages improves adaptability and interoperability with different help apparatuses.

Python's solid library support for AI training further underscores Python's appropriateness for this venture. Striking libraries incorporate NumPy and SciPy for mathematical and logical registering, Pandas for information examination, Scikit-learn for improved information investigation and demonstrating, and the TensorFlow and PyTorch team — Profound learning structures assist with smoothing out the advancement of brain network models. Of specific importance, the PyTorch library will assume a vital part in the impending phases of this task.

At its center, Python's effortlessness, clarity, tremendous library, and solid local area support have situated it as the quintessential language for both training and conveying AI models. Its versatility empowers designers to participate in an assortment of AI applications, spreading over from ordinary AI to advanced profound learning endeavors. Subsequently, picking Python as the main programming language for this undertaking isn't just sensible but additionally savvy because of its innumerable advantages and materialness.

3.3.2 Anaconda

Anaconda is an open-source bundle for the executives and climate of the board framework for programming improvement. Clients utilizing Anaconda can without much of a stretch make do, introduce, and coordinate programming bundles or set up conditions that are totally free and isolated from the base framework and climate. Anaconda is well known in the fields of information, profound learning, AI, and logical registering in light of its capacity to oversee complex programming conditions. In this venture, I will involve Anaconda in overseeing libraries and programming conditions for the task and establish and oversee conditions well defined for this undertaking.

3.3.3 PyTorch

PyTorch is an open-source AI library created by Facebook's AI Exploration lab. Known for its adaptability and easy-to-use plan, PyTorch is generally utilized for building and training profound learning models. Its dynamic computation chart takes into account quick changes, making it simple to create and troubleshoot models. PyTorch works on tensors, supporting effective calculation and GPU speed increase. With its Pythonic approach, it offers an instinctive programming experience. PyTorch's lively local area and huge environment incorporate libraries, devices, and pre-trained models. Deployability and profound reconciliation with research settle on PyTorch a well-known decision for both scholarly exploration and reasonable applications in AI. Not a special case, my task will likewise utilize the PyTorch library to help AI training with Model YOLOv8.

3.3.4 PyQt5

PyQt5 fills in as a Python interface to the Qt application structure, permitting the production of work area applications with a graphical user interface (GUI). It incorporates a bunch of Python modules that embody Qt classes, working with the smooth joining of Qt usefulness into Python-based applications. Perceived for its effortlessness, user-benevolence, and expansive similarity across Windows, macOS, and Linux stages, PyQt5 is a flexible instrument for designers. Its broad toolset incorporates different GUI parts, design supervisors, and occasion overseers, engaging engineers to exploit the rich list of capabilities. Broadly utilized for some applications, PyQt5 obliges everything from straightforward utilities to complex programming requiring complex user interfaces. The mix of Python's capacities with the strong Qt structure pursues PyQt5 a well-known decision for GUI improvement. Thus, I decided to involve PyQt5 as the fundamental part of this undertaking, to make a GUI that is user-accommodating and simple to explore.

3.3.5 LabelImg

LabelImg stands apart as an open-source device intended to make object recognition datasets effortlessly. Made by Tzutalin, the device flaunts an easy-to-use interface that works with a consistent drawing of jumping boxes around items and tasks of names. It is versatile and upholds different comment designs, improving the dataset readiness process for preparing models like YOLO. Its easy-to-use configuration takes special care of the two fledgling and experienced scientists, adding to the productivity of PC vision and AI endeavors. With a functioning client local area and constant enhancements, LabelImg keeps up with its crucial job as a primary device for picture explanation in the field of item identification. That is the reason I chose to involve LabelImg as the application to comment on the information pictures used to prepare the YOLOv8 model.

3.4 Overall

The methodology employed for developing an AI solution dedicated to recovering lost items in educational settings is comprehensive and meticulously outlined in Chapter 3. The approach prioritizes transparency and replicability, addressing technical, ethical, and user-centric considerations. The methodological components include study design, data collection, YOLOv8 model training, system implementation, assessment metrics, user interface design, and data

analysis. Notably, the research design encompasses learning and researching knowledge, designing an implementation plan, project implementation, and result evaluation.

The YOLOv8 architecture, dissected into Backbone, Neck, and Head components, demonstrates a sophisticated model structure for object detection. The model training process involves data augmentation, dataset splitting, grid division, computing losses, balancing losses, eliminating grid sensitivity, and building targets. The YOLOv8 architecture's unique features, such as anchor boxes and modified equations for predicting bounding box coordinates, contribute to decreased grid sensitivity and improved model performance. The detailed exploration of these elements reflects a robust methodology, ensuring efficient training and accurate object detection in the AI solution.

Finally, I use Python as the main programming language as well as take YOLOv8 as a model for development. In addition, I use other supporting applications and software for the project such as Anaconda to support creating virtual environments, LabelImg to support annotate, libraries such as PyTorch, PyQt5, ... All the above will be used. Use and support me to carry out this project.

CHAPTER 4: IMPLEMENTATION AND RESULTS

Chapter 4, "Implementation and Results," outlines the meticulous deployment of the YOLOv8 model for object detection in an educational context. It provides a detailed guide on data collection, preparation, virtual environment setup, and model selection, emphasizing the thoughtful consideration of hardware limitations. The chapter delves into critical training parameters such as epochs, model types, and image size, ensuring a comprehensive understanding. The addition of a user-friendly interface for diverse users enhances accessibility. The project's success is evident in its ability to accurately detect objects, even in video files. Overall, Chapter 4 offers a thorough and insightful walkthrough of the practical implementation process.

4.1 Data Collection

4.1.1 Selection of Dataset

For the training and assessment of the YOLOv8 model, it is essential to curate a thoughtfully chosen dataset. This dataset should accurately reflect the common items that students frequently misplace in educational settings, encompassing items like computer mice, chargers, phones, wallets, calculators, and more. It is crucial to account for the diverse range of items that might be lost and anticipate potential challenges, including obstructions, shooting angles, and fluctuations in lighting conditions.

Nevertheless, due to the current hardware limitations of computers, continuously expanding the repertoire of items recognizable by AI poses challenges. Adding each item to the system may require extensive retraining, leading to prolonged durations or potential issues with system memory constraints. Hence, in the Data Collection section, we adopt an approach of categorizing data into distinct fixed types, sub-dividing it to prevent the dataset from becoming excessively large and impacting system performance.

This rationale guided my experimentation, focusing on two prevalent types of objects in the research (When put into practice, it is possible to run many other objects in parallel): calculators and wallets. For calculators, I introduced two models with nearly identical shapes to pose a challenge for the AI in recognizing intricate details and distinguishing between them accurately. Additionally, three wallets of different sizes, shapes, and colors were utilized as training data, aiming to ensure that the AI can recognize diverse objects of the same type without displaying bias.

To train and evaluate the YOLOv8 model, it is essential to curate a carefully selected dataset. This data set should accurately reflect common items that students frequently misplace in educational settings, including items such as computer mice, chargers, phones, wallets, calculators, and more. It's important to take into account the variety of items that can be lost and anticipate potential challenges, including obstructions, shooting angles, and fluctuations in lighting conditions.

However, due to current hardware limitations of computers, continuously expanding the catalog of items recognized by AI poses challenges. Adding each item to the system may require extensive retraining, leading to extended durations or potential issues with system memory constraints. Therefore, in the Data Collection section, we apply the method of classifying data into separate fixed types, breaking it down to prevent the data set from becoming too large and affecting system performance.

This rationale guided my testing, focusing on two common types of objects in research (When put into practice, many other objects can be run in parallel): computers and wallets. For the computer, I introduced two models with nearly identical shapes to challenge the AI to recognize complex details and accurately distinguish between them. Additionally, three wallets of different sizes, shapes, and colors were used as training data to ensure that the AI can recognize different objects of the same type without displaying bias.

Figure 9 The image of the object is taken from the front

Figure 10 Photo of all 3 common objects mixed with many other strange objects

At the same time, I have gone through many experiments and come up with a common denominator for each object that needs to be trained to optimize the time and resources used to train each item and still achieve the best results. For each object, the trainer needs to prepare a total of AT LEAST 25 images which are classified. 18 photos will be taken separately of that object with many different backgrounds and lighting. Then it will be divided into 2 parts, 15 panels will be put into the Training set to train the model and 3 panels will be put into the

Validation set to be used as comparison parameters. Similar to the object with the remaining 7 pictures, the object will be taken with the same objects to be classified and trained together, in which 2 pictures will be included in the Validation set and 5 pictures will be included in the Training set.

Table 1 Statistics on the size of each object dataset

Object	Images of only that object in train set	Images of only that object in validate set	Images of only that object in test set	Images of that object mixed with other objects in train set	Images of that object mixed with other objects in validate set	Images of that object mixed with other objects in test set
calculator1	20	6	7	14	3	7
calculator2	11	3	7			
wallet020923	14	1	3	13	2	1
wallet180923	10	1	3	13	2	
wallet051023	11	1	15	15	2	

**The data in the merge cells refers to all the objects in the mix objects images that are in the same image.*

Table 2 Statistics on the size of each object type dataset

Object Type	Train set	Validate set	Test set
Calculator	45	12	21
Wallet	54	6	22
Total	**99**	**18**	**43**

4.1.2 Data Preparation

Prior to inputting the data into the model, a set of preprocessing procedures is executed. This involves cleaning the data, addressing missing values, introducing random factors as detailed in the Research Design section, and standardizing formats. The objective is to guarantee that the dataset is systematically organized, devoid of irregularities, and prepared for optimal model training.

The first is to use the camera to record video and take photos needed to train AI. However, due to hardware limitations and research time, it is necessary to limit images to only be taken from a top-down angle to facilitate AI training. In addition, to diversify the input data, the images must also be taken in different backgrounds. There must also be photos with different light densities and sharpness to increase the AI's ability to recognize objects in even imperfect conditions.

Figure 11 Use the camera to take pictures of objects

Figure 12 Use the camera to take photos of objects at low brightness

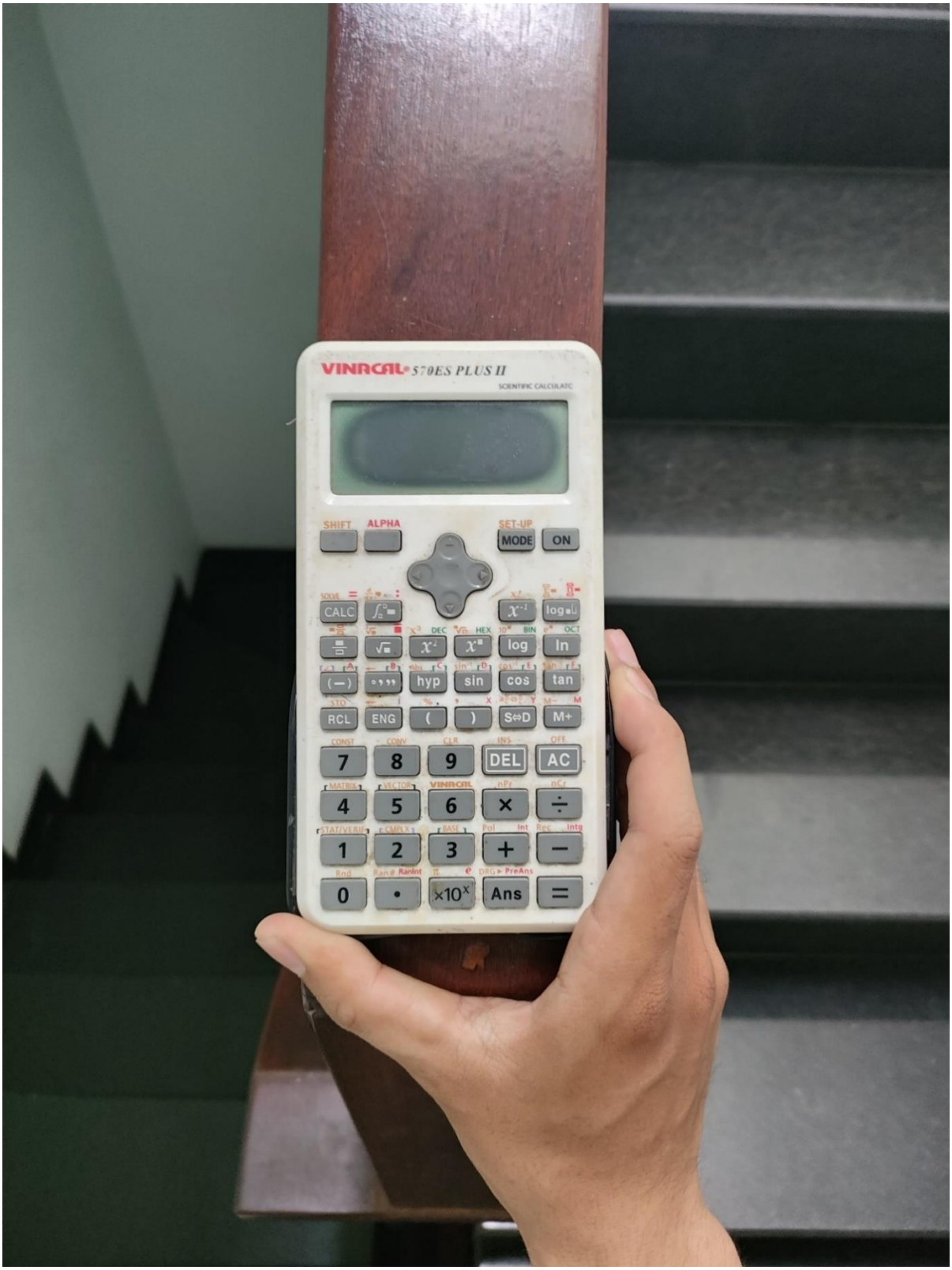

Figure 13 Use the camera to take photos of objects in many different backgrounds

Then, carefully select input photos that meet the standards. Choose enough input images to train the AI, not too few to reduce the model's accuracy, but not too many to waste resources and time when training the AI. Finally, synthesize all input data and divide it into 2 types of datasets: Training set and Validation set (mentioned and analyzed in part 3.3.2.2 Splitting Data Set). With this division, when AI trains the data in the Training set, it will use the data frame in the Validation set to evaluate the performance, accuracy and reliability of the current data and then calibrate it to achieve the results. best results.

4.1.3 Setup Virtual Environment

In software development and data science, establishing a controlled and segregated workspace for project tasks is crucial. The virtual environment functions as an enclosed area where project-specific dependencies, libraries, and configurations can be independently managed, separate from the broader system environment. This practice ensures uniformity across diverse projects and mitigates conflicts between dependencies. This guide delves into the procedures for configuring a virtual environment, presenting a systematic approach to achieving isolation, managing dependencies, and ensuring project reproducibility. Whether you are engaged in Python application development, data analysis, or machine learning model deployment, the establishment of a virtual environment is a foundational step toward constructing a reliable and reproducible development environment.

To install and create a project with anaconda just run this code “conda create -n name_of_project”. Then run “conda activate name_of_project” to activate a specific anaconda environment for the next steps. Then create a folder “Images” to store the input data for training AI.

4.1.4 Annotate with LabelImg

Accurate ground truth annotation is paramount for training the YOLOv8 model effectively. Each image in the dataset is meticulously annotated to indicate the location and class of lost items. To enhance accuracy, a robust annotation process is followed, incorporating multiple annotators and verification steps. The process aligns with established best practices for training object detection models.

First to install labelImg run the following command from Command Prompt "pip install labelImg". After installation is complete, just type the command "labelImg" to launch the program. Then to annotate, click the "Open" button on the left side of the interface to select the folder containing the data previously saved to annotate:

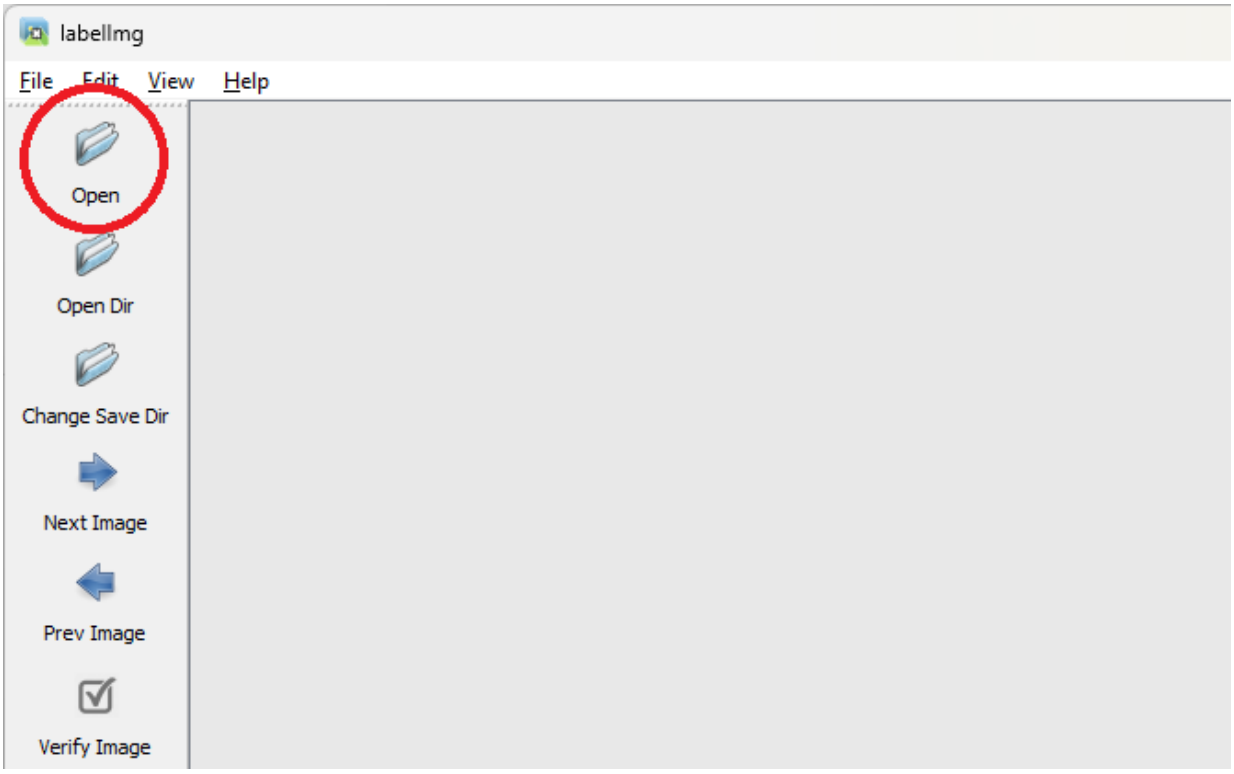

Figure 14 Open button of LabelImg

Next select File -> Change save dir to change where the output files will be saved to determine the location and coordinates of the object that needs to be annotated on the input image. Remember to create a "labels" folder next to the "images" folder to hold these note files:

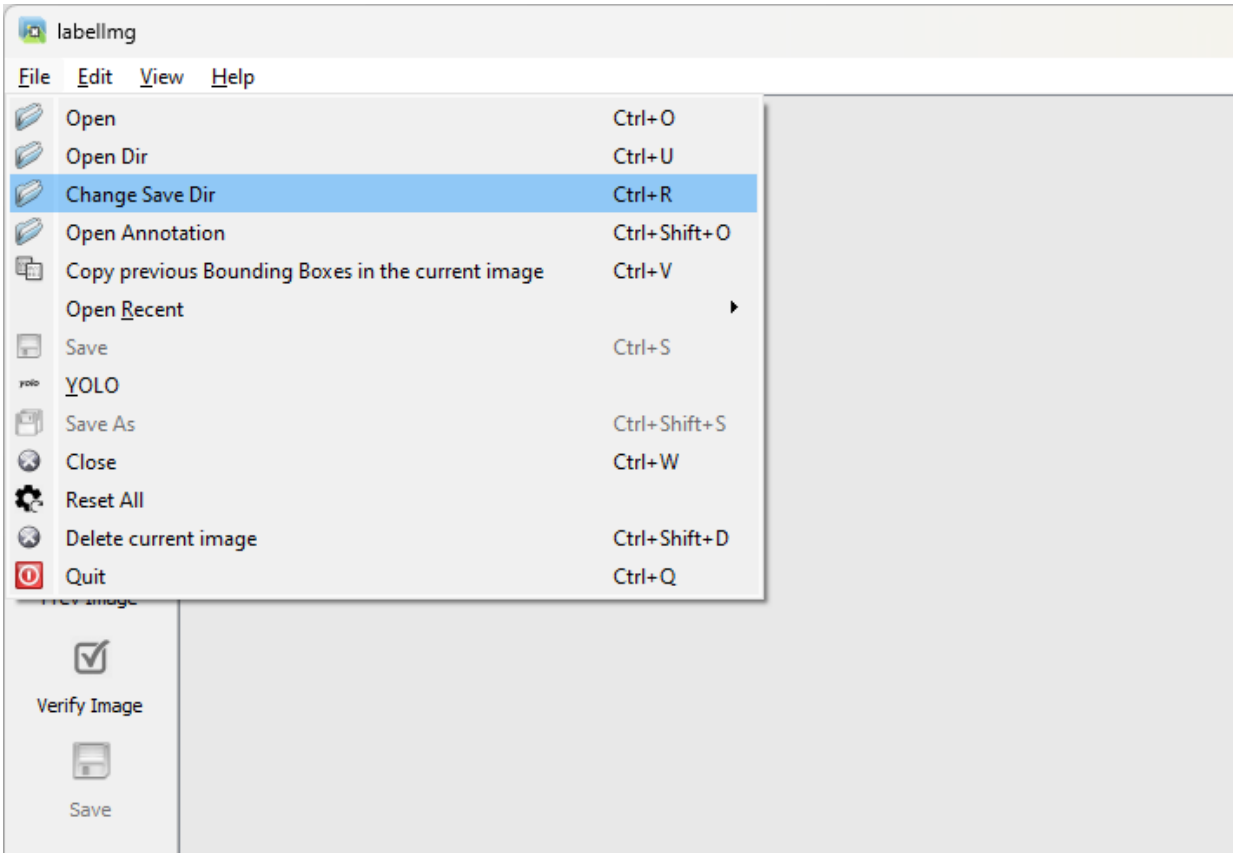

Figure 15 Change Save Dir

Don't forget to adjust the Output format to Yolo model to be compatible with the model I will use:

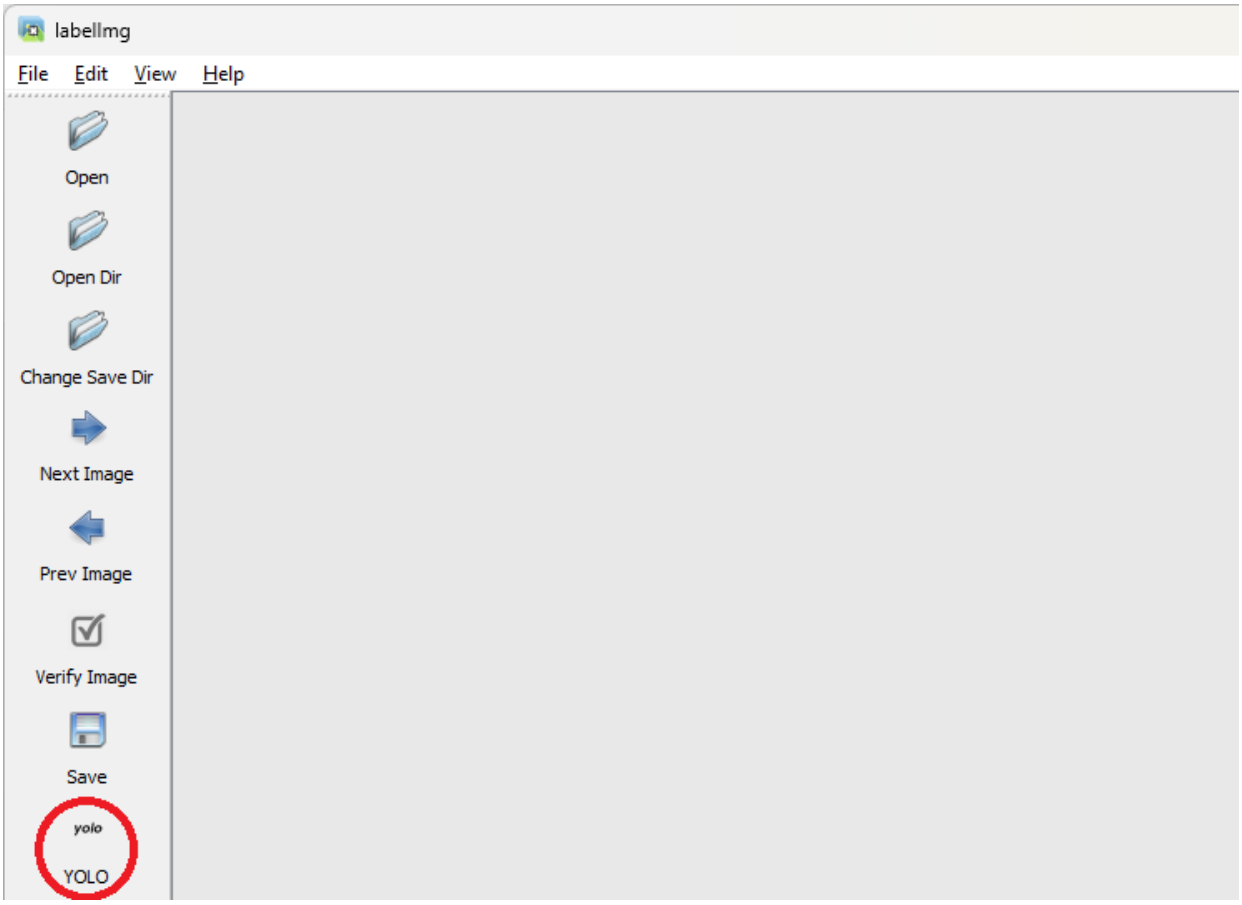

Figure 16 Choose ouput format

Then just click the "Create RectBox" button to start positioning objects on the input image:

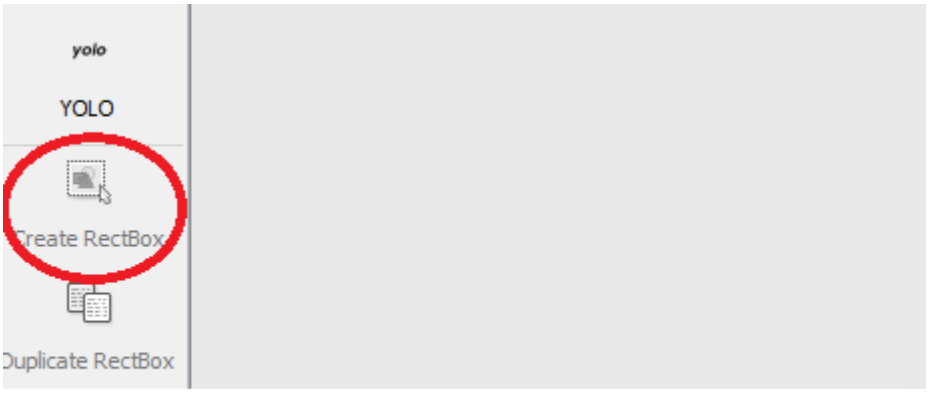

Figure 17 Create Detect Box Button

After locating, Enter the name of the object being annotated to distinguish it. After annotating, it will output 2 files inside the label folder. 1 file will be the coordinates used to

position the annotated objects in the corresponding image file. The remaining file will be classes.txt to declare annotated objects in all files.

Figure 18 The IMG20231008143828.jpg file

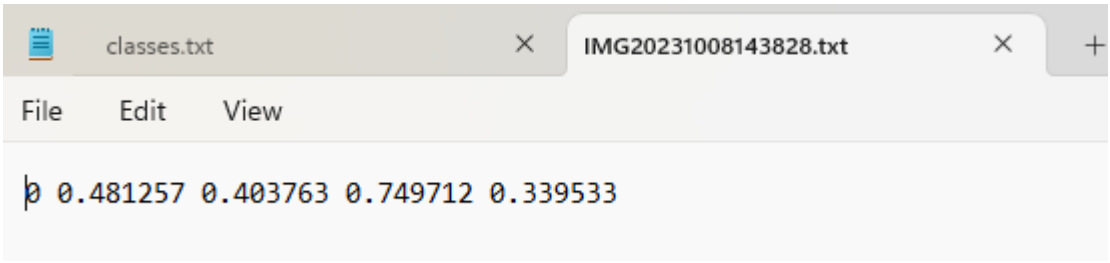

Figure 19 The output file of IMG20231008143828.jpg

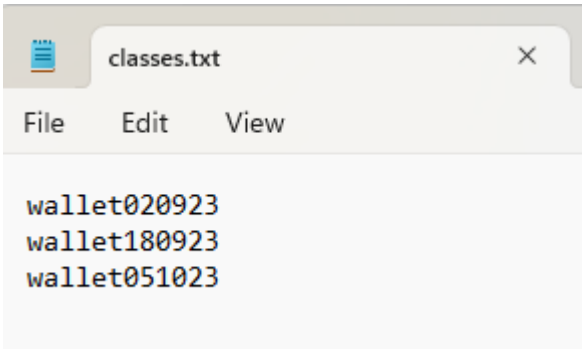

Figure 20 class.txt

Finally, a .yaml file is needed to configure information related to training a YOLOv8 model. This file needs to declare the Training Directory, Validation Directory (I will mention it in the algorithm analysis section later), Number of Classes, and Class Names.

Figure 21 data_custom.yaml

4.2 Setup YOLOv8 repository

Setting up the YOLOv8 repository is a crucial initial stage in deploying YOLO object detection on a custom dataset. This repository serves as the essential foundation, offering the

required codebase and tools to facilitate the training and deployment of YOLOv8 models. Ultralytics stands out as a prominent open-source deep learning research entity, celebrated for its key library. Focused on computer vision and machine learning, Ultralytics offers an intuitive interface, streamlining activities such as training and deploying advanced models. This library, compatible with well-known frameworks including PyTorch, is acknowledged for its ease of use and engaged community, serving as a valuable asset for professionals and researchers in the artificial intelligence domain.Run this command on command prompt “pip install ultralytics” to install the Ultralytics library using pip.

PyTorch for CUDA 11.7, commonly referred to as cu117, is an optimized variant of PyTorch designed to leverage the GPU acceleration provided by NVIDIA's CUDA toolkit version 11.7. This customized release exploits the capabilities of GPUs compatible with CUDA to notably enhance the speed of deep learning computations. Run “pip3 install --upgrade torch torchvision torchaudio --index-url https://download.pytorch.org/whl/cu117” to upgrade PyTorch, torchvision, and torchaudio to the latest version specifically for CUDA 11.7 (cu117).

4.3 Training model

This is also the most important step in implementing this project. In the previous steps, we have initialized the necessary frameworks for training AI such as preparing datasets, annotations, setting up the environment, etc. However, in order to be able to train a model with optimization for the with current hardware, training time as well as the accuracy and breadth of AI's object recognition, we need to determine the following optimal parameters:

4.3.1 Epochs

The YOLOv8 model undergoes training for a specific number of epochs, a hyperparameter indicating how many times the entire training dataset is processed through the neural network in both forward and backward passes. In each epoch, the model parameters are adjusted through backpropagation, using computed gradients.

Determining the optimal number of epochs involves considering factors such as dataset complexity, model architecture, and the convergence patterns during training. It's a common practice to assess model performance on the validation set throughout training to decide when to conclude training and prevent overfitting.

Consequently, optimizing the number of epochs is crucial. Following numerous iterations of AI training and comparative analysis (covered in greater detail in section 4.4), I've identified the suitable number of epochs based on the chosen dataset template. Specifically, I've set it at 80 epochs if the training involves fewer than 10 objects and fixed it at 100 epochs for cases with more than 10 distinct objects. While this might seem extensive initially, the need for precision in detecting items from varied angles and lighting, coupled with the challenge of distinguishing similar objects, justifies the selection of 80-100 epochs as the most optimal for training this model.

4.3.2 Model

YOLOv8 model is divided into 5 different model types with accuracy and parameters I researched and calculated as follows:

Table 3 Compare parameters of YOLOv8 models

Model	**size (pixels)**	$\boldsymbol{mAP^{val}}$ **(50-95)**	**Speed CPU ONNX (ms)**	**Speed A100 TensorRT (ms)**	**params (M)**	**FLOPs (B)**
YOLOv8n	640	37.3	80.4	0.99	3.2	8.7
YOLOv8s	640	44.9	128.4	1.20	11.2	28.6
YOLOv8m	640	50.2	243.7	1.83	25.9	78.9
YOLOv8l	640	52.9	375.2	2.39	43.7	165.2
YOLOv8x	640	53.9	479.1	3.53	68.2	257.8

In that:

Size (Pixels):

Describes the dimensions of the input images in pixels that the YOLOv8 model is trained and assessed on. This measurement is essential for comprehending the image resolution processed by the model.

$\boldsymbol{mAP^{val}$(50-95):

Denotes mean Average Precision (mAP) across various IoU (Intersection over Union) thresholds spanning from 50% to 95%. mAP is a widely-used metric for assessing the precision of object detection models, providing an average precision value across diverse IoU thresholds.

Speed CPU ONNX (ms):

Reflects the inference speed of the YOLOv8 model on a CPU utilizing the ONNX (Open Neural Network Exchange) format. It signifies the average time taken for the model to analyze an input image and generate predictions, measured in milliseconds.

Speed A100 TensorRT (ms):

Represents the inference speed of the YOLOv8 model on an A100 GPU using TensorRT (NVIDIA's high-performance deep learning inference library). Similar to CPU speed, this metric indicates the time required for inference, measured in milliseconds.

Params (M):

Denotes the quantity of parameters in the model, typically expressed in millions (M). It characterizes the model's size in terms of learnable weights and biases.

FLOPs (B):

Abbreviation for Floating Point Operations Per Second, measured in billions (B). This metric gauges the computational workload of the model during inference, with higher FLOPs values generally indicating increased computational complexity.

However, due to hardware limitations, we cannot use models that are too complex, which will lead to some errors when training AI and even not enough memory to run training. However, if you use a model that is too simple, it will result in the model not being able to recognize items after training or requiring a high number of epochs to achieve good results. This leads to a waste of resources and time when training AI.

Let's look at the parameter $\boldsymbol{mAP^{val}}$ **,** which represents mean Average Precision across various IoU thresholds. This is an extremely important parameter, the higher the data, the higher the accuracy of that model. Among the 5 experimental models, the variation from YOLOv8n to YOLOv8s is the highest with 7.6 (from 37.3 to 44.9). At the same time, that value from YOLOv8s to YOLOv8m still increased a lot with 5.3 (from 44.9 to 50.2). From YOLOv8m to YOLOv8l and YOLOv8x the increase is very small (2.7, 1.0). Furthermore, the other parameters and complexity of YOLOv8m are completely compatible with most types of hardware on popular computers or even laptops today. Therefore, after considering all the data and details, I decided to use the YOLOv8m model for training.

4.3.3 Imgsz

When training a YOLOv8 model, the term "imgsz" commonly denotes the dimensions or resolution of the input images throughout the training procedure. More precisely, it indicates the width and height of the input images measured in pixels. However, if imgsz is too small, it will easily lead to loss of information or the information becomes too blurry and difficult to receive, causing model training efficiency to be significantly reduced. After researching and evaluating, I chose the imgsz = 640 parameter as the best parameter because it is large enough to represent finer details for the input, presenting a larger image to avoid losing small details but However, it is still enough to optimize training resources. Because objects often have almost the same shape, the parameter 640 is necessary to represent small details but is crucial for the model to be able to distinguish between different objects.

4.3.4 Batch

In the process of training the YOLOv8 model, the term "batch" signifies the quantity of images handled in each training iteration. This batch size serves as a hyperparameter, dictating the number of samples, or images, processed by the neural network before updating model parameters based on computed gradients. Larger batch sizes often yield more stable and efficient training, leveraging parallelism in modern hardware like GPUs. However, the trade-off involves increased memory requirements, and excessively large batches may exhibit diminishing returns in terms of accuracy. After assessing my laptop's GPU capabilities, I found that a batch size of 8 is optimal. Nevertheless, when utilizing different computers, such as those in a school office, this number may be adjusted accordingly.

4.3.5 Patience

In the YOLOv8 model training context, the term "patience" is commonly linked with the early stopping technique. Early stopping is a regularization method designed to prevent overfitting, wherein training halts if the model's performance on the validation set plateaus or worsens over a specified number of consecutive epochs. The "patience" parameter dictates the threshold for the number of epochs without improvement in validation performance that the training process will endure before ceasing. If the model fails to show improvement over the consecutive epochs determined by the "patience" value, training is halted to avert overfitting and conserve computational resources. Following extensive research and experimentation, the optimal patience value is established at 128, reducing the maximum number of epochs the system runs each training session by 10-20%, resulting in a more optimized system and resource efficiency.

4.3.6 Worker

In the YOLOv8 model, the parameter labeled as "worker" is intricately linked to the quantity of data-loading processes employed in the training phase. This parameter plays a pivotal role in the data loading mechanism, specifically dictating how training data is fed into the model in each iteration throughout the training process. The "worker" parameter serves as the determinant for the number of parallel processes or threads dedicated to the loading and preprocessing of data. Elevating the number of workers can expedite the data-loading process, particularly when dealing with extensive datasets. This approach facilitates the concurrent loading and preprocessing of multiple data batches, leveraging the capabilities of multi-core CPUs. Nonetheless, an excessive number of workers may lead to congestion issues, resulting in insufficient memory for program execution and consequently triggering an error:

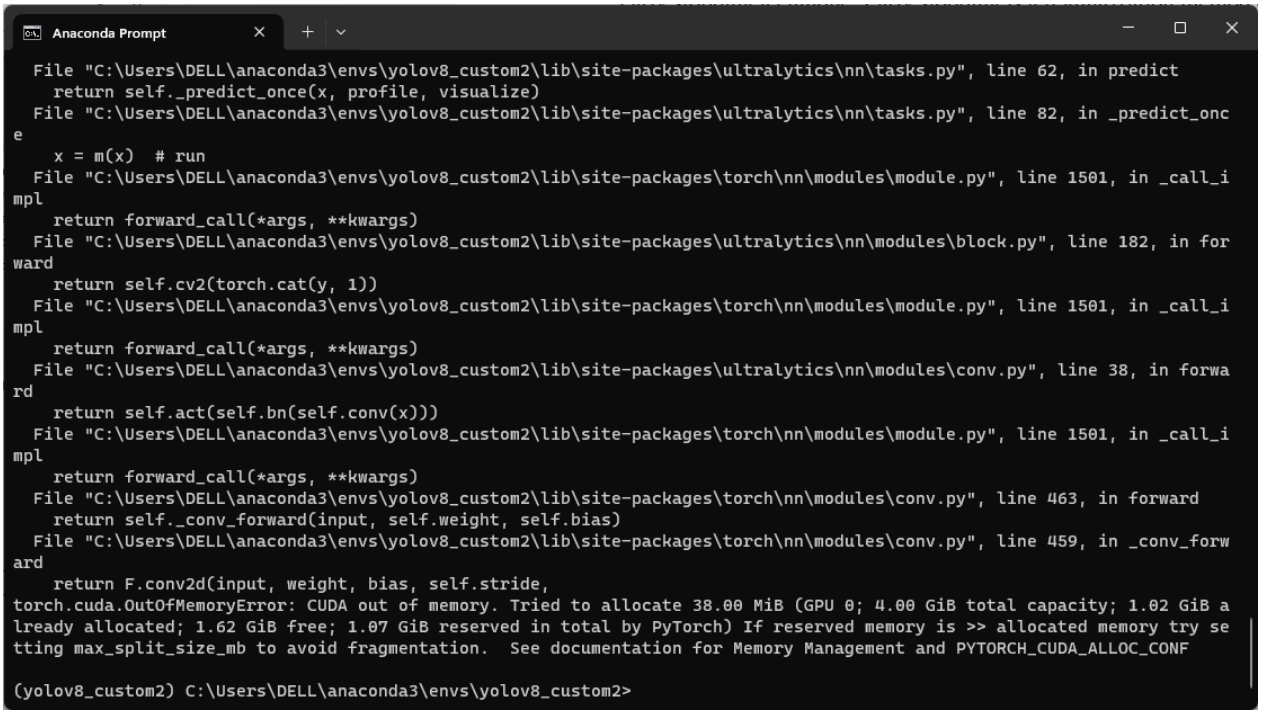

Figure 22 Out of memory when workers > 1

Due to hardware limitations, my project can only run at worker =1.

4.4 Building UI

However, I realize there is an inconvenience because the target audience for this project is all IU students. That means there are students in fields that have nothing to do with computers, so it will be very difficult to use the command line to run this project. That's why I designed a UI so that students can use it more easily. The project's UI is written with the PyQt5 library.

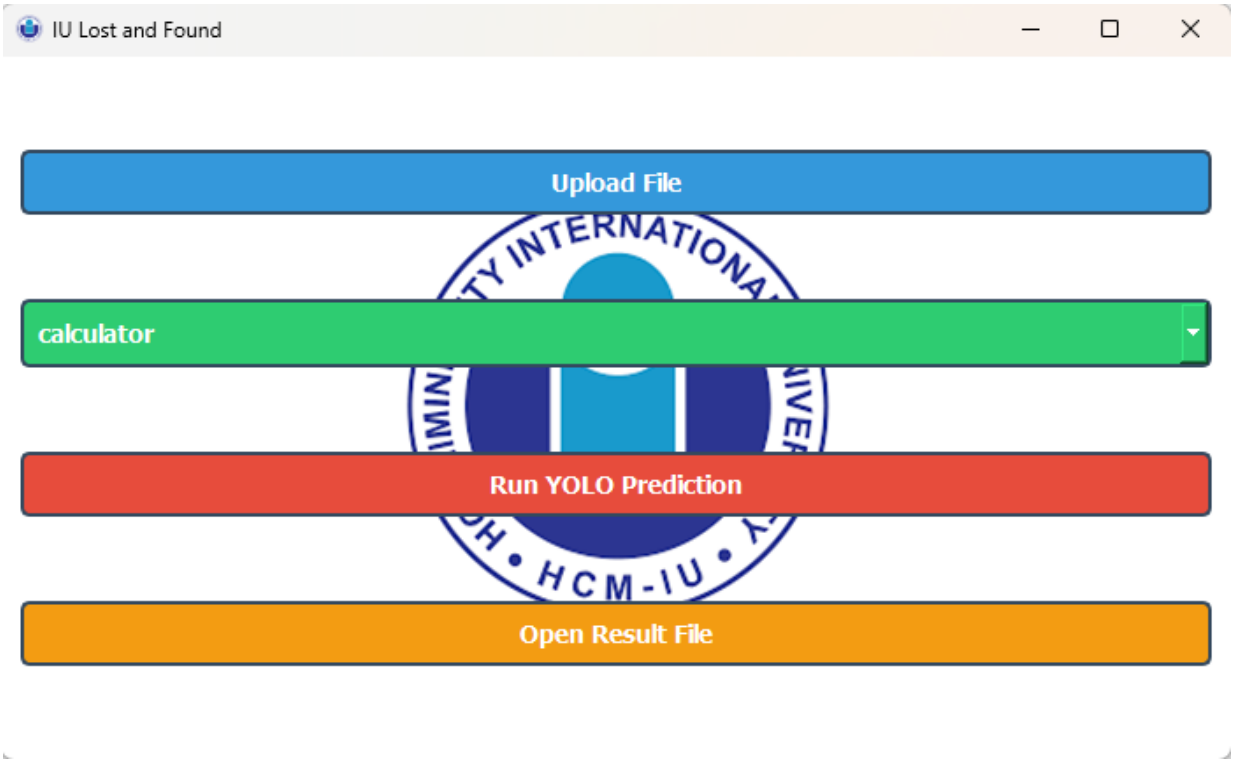

Figure 23 UI

4.4.1 Design

To initiate, I will delineate the components integral to my User Interface (UI) to facilitate seamless user interaction. Commencing with a button feature, users can effortlessly upload an image of the lost item, providing essential input data for the model. Following this, given the diverse categorization of the dataset, a selection button will be incorporated, enabling users to specify the type of object they are seeking. Subsequently, a button will be implemented to trigger the initiation of the object detection process by the model. Lastly, a button will be featured to unveil the output file, allowing users to ascertain whether the object has been located within the student affairs room.

In parallel, it is imperative to enhance the visual appeal and clarity of the interface. Simultaneously, instructional cues will be seamlessly integrated into the UI, guiding users through the process and mitigating the likelihood of errors. This comprehensive approach aims to create an engaging and user-friendly interface, ensuring users can navigate through the functionalities effortlessly and understand the procedure at every step.

4.4.2 Upload file button

In the Upload file button, I use the file_path variable to get the name and file type of the uploaded file. Then I use an if statement to check whether there is a file uploaded or not. If there is, the system will move that file inside the project's main directory so that the model will later use it to detect the object.

4.4.3 Dropdown button

To be able to launch the model, it is necessary to determine the type of object the user wants to detect. When clicking on the button, names of object types will appear, and the user will select the corresponding object type. Later, when needed, just use the self.model_dropdown.currentText() structure to call the necessary string.

4.4.4 Run YOLO Prediction button

This button has the effect that when clicked, it will start launching the model to detect objects. First of all, to run the model we need to start the anaconda environment. I used os.environ to set anaconda's environment variables and then started the environment. Next, I used self.latest_uploaded_file to get the file path of the input file. Next, use an if statement to check whether there is a file uploaded or not. If so, use the variable selected_model = self.model_dropdown.currentText() to get the name of the item type that the model needs to detect from the dropdow button. Then just use the command to run the command line "yolo_command = f'yolo task=detect mode=predict model={selected_model}.pt show=True conf=0.5 source={uploaded_file}' ". At the same time, I also insert notification sentences when the model is detecting so that the user knows that the system is running and waiting for results. Another message will appear after the model has been detected.

4.4.5 Open result file

This button has the effect that when the user presses it, it will open the resulting image after detection is complete. To do so, there will be a result_folder variable that refers to the folder containing the result file. Then the algorithm I built will take the time of all the files in the folder and select the newest file then open it for the user to see.

4.4.6 Usage Procedure

First the user will click on “Upload file” button and select the file containing the image of the object to be searched to upload to the system:

Figure 24 Upload File Button

The user then clicks on the dropdown button below and selects the type of object they are looking for:

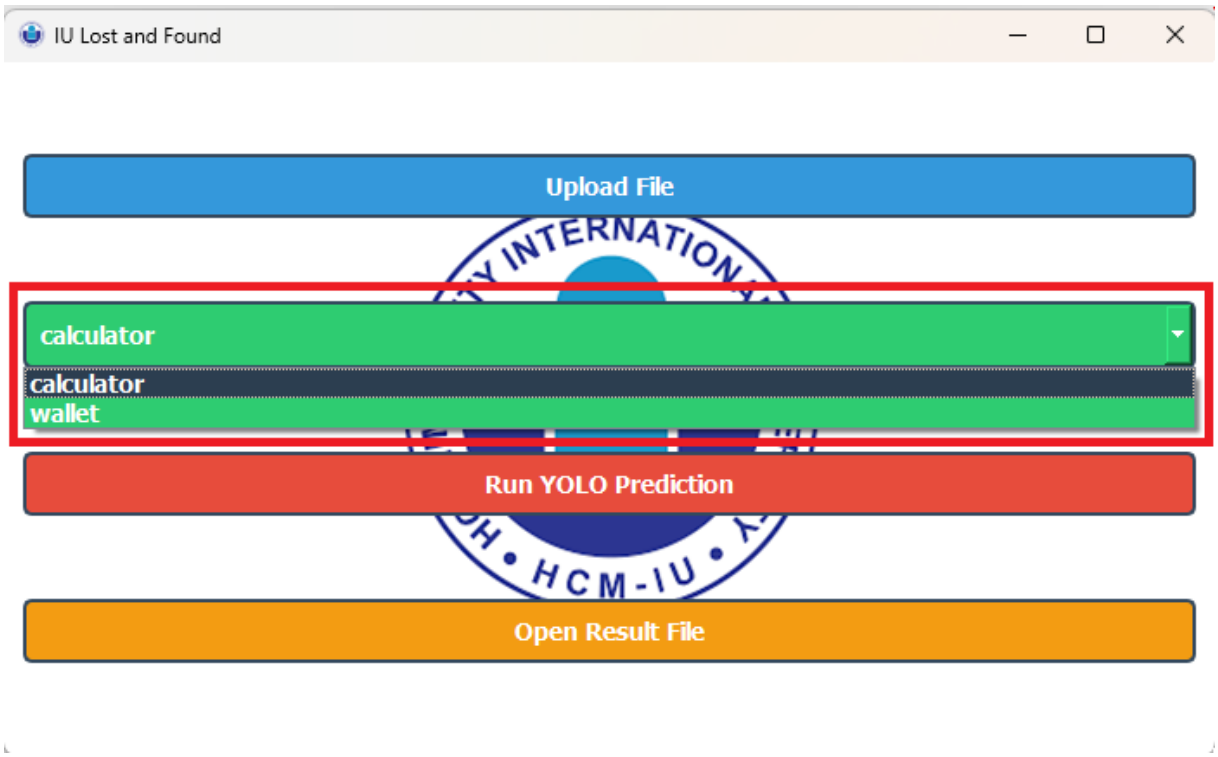

Figure 25 Select object button

The user then presses the "Run YOLO Prediction" button, and a message appears to indicate that the system is detecting.

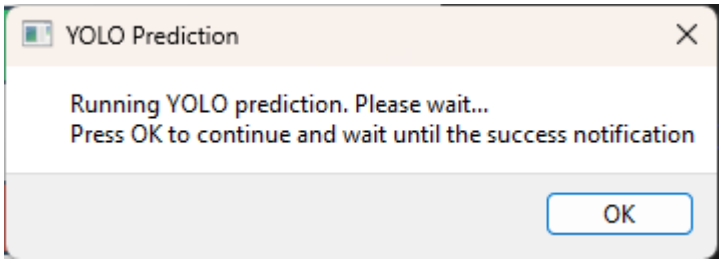

Figure 26 Running prediction notification

The user clicks ok and waits for the notification that the model has been detected.

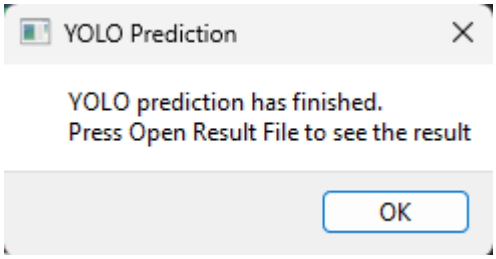

Figure 27 Prediction finished notification

Finally, the user clicks the "Open Result file" button to open the file to view the results.

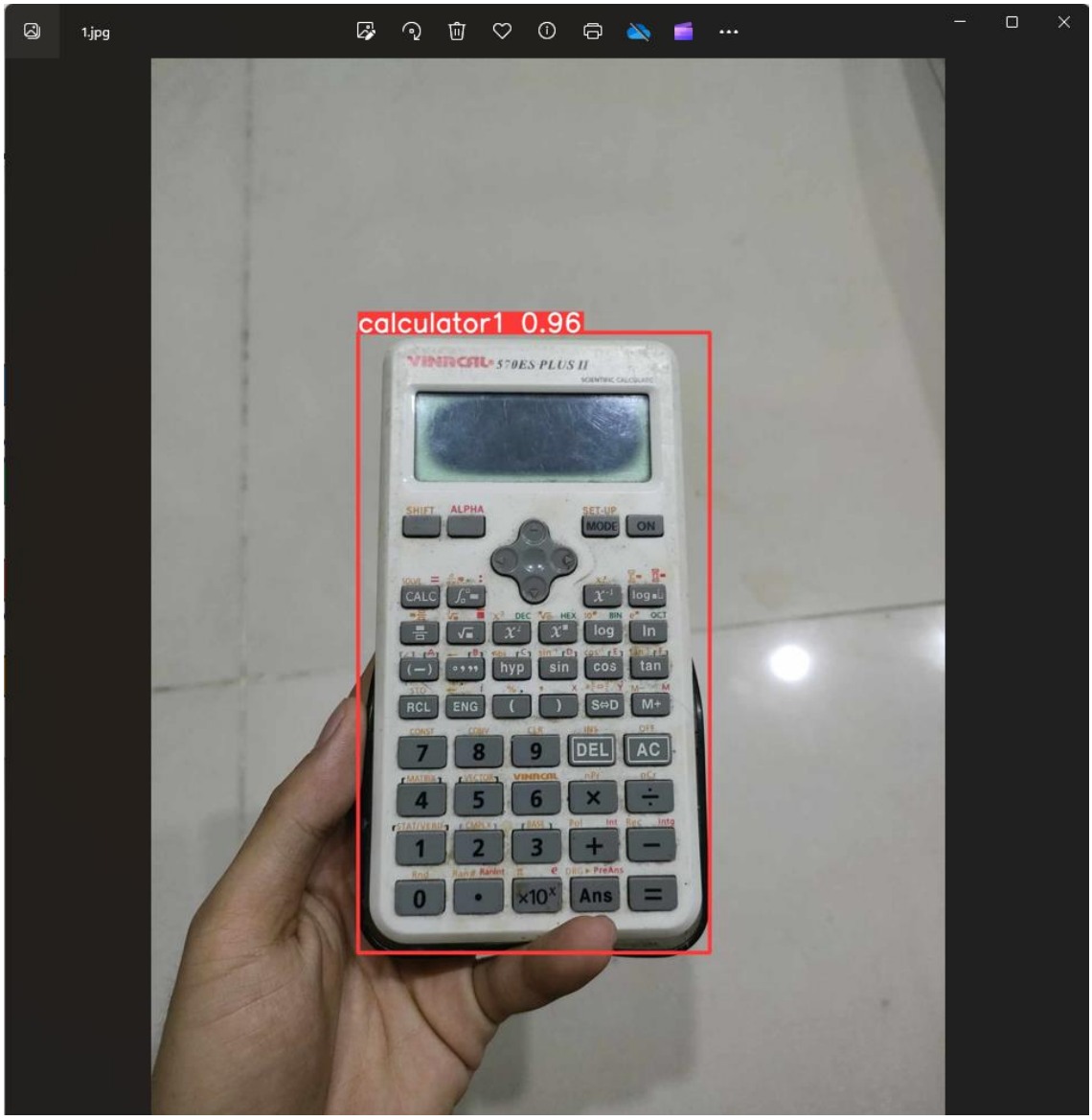

Figure 28 Result file

4.5 Detect Object Result

After training the model, we will get an output folder and will use it to detect trained objects in images or videos uploaded by users. First, we need to pay attention to the weights, these are the output parameters of the training process, and it will be used as a template during the detection process. After training is complete, we will obtain 2 sets of weights: best.pt and last.pt. The best.pt file represents the parameters that provide the best performance throughout the entire training process. The "last.pt" file contains the model parameters from the final training epoch. Therefore, I would recommend using the best.pt file as the main parameter for the detection process. To be able to detect, from the Project environment, we need to run the

following command line "yolo task=detect mode=predict model= best.pt show=True conf=0.5 source= 1.mp4". In that source is the input file name.

After launching the command, the system will begin dividing the input image data into cells in grid and use the parameters of best.pt to start detecting objects according to the algorithms mentioned in chapter 3. The model will use a bounding box to mark the object in the image with properties such as height, width, class name, center of the box (b_x, b_y). In this project, I was able to train the model to recognize items of similar size and shape. At the same time, it is not only limited to image files, but video files are also extremely accurate. Even the model will not be confused with items that have not been trained.

Figure 29 Detect different wallets in same type

Figure 30 Detect different objects in same type

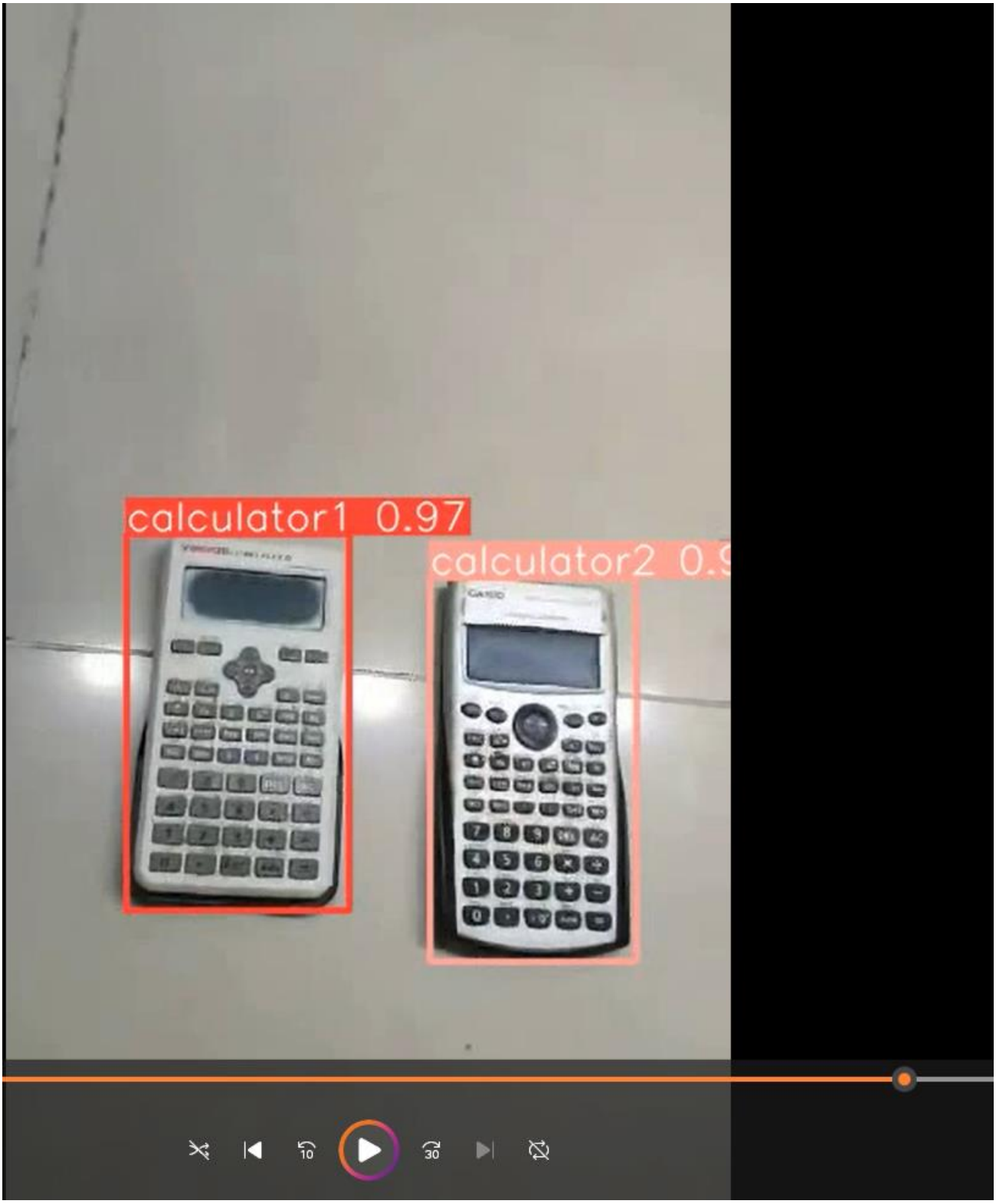

Figure 31 Detect in video

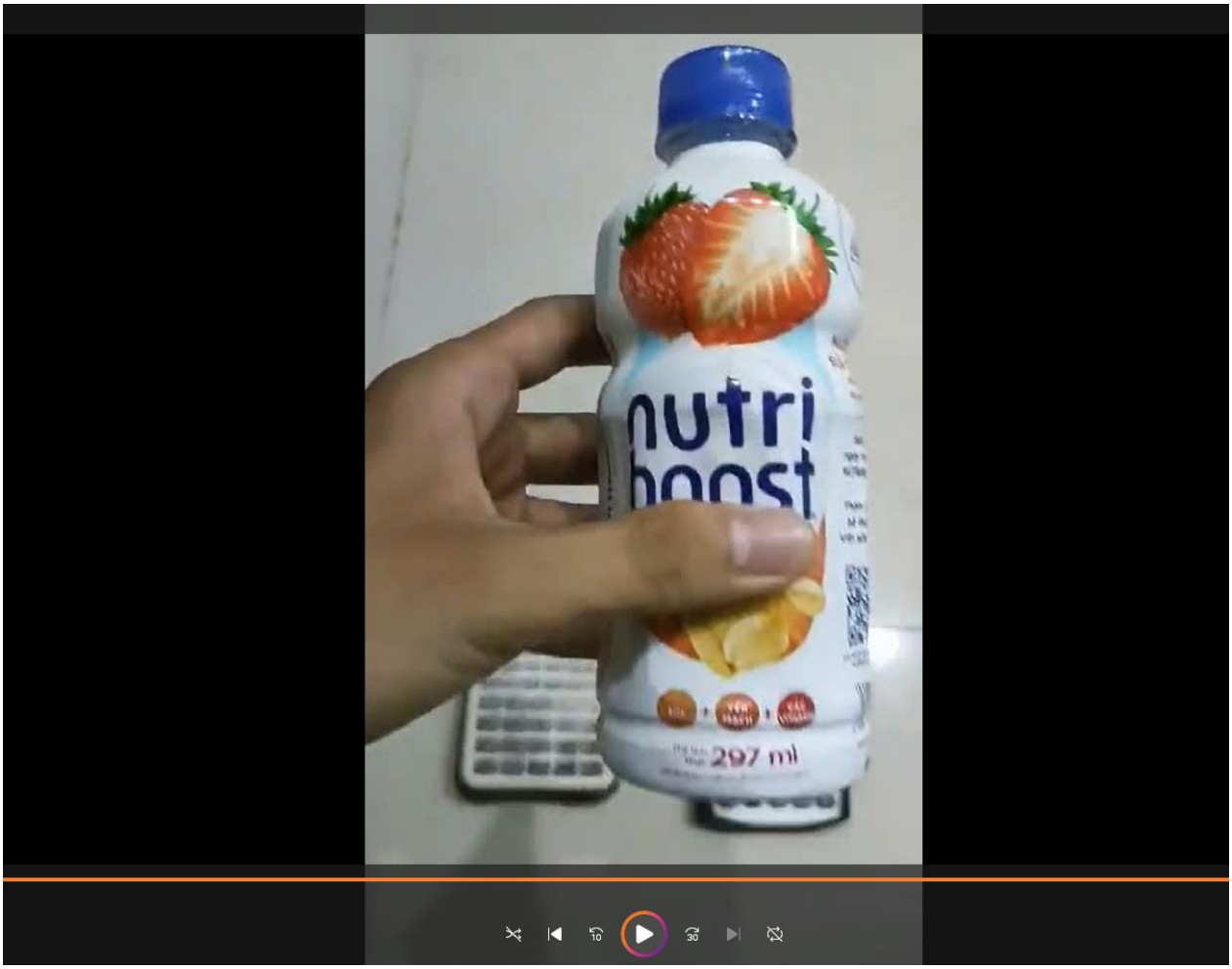

Figure 32 Not mistakenly detected as a foreign object

4.6 Overall

In short, I successfully completed my project. By improving and Splitting the dataset, I have improved the input data for the train model. After that, I researched and found the most optimal solution for training the model and this helped my model achieve much better results. My model can distinguish objects of similar shape and size most accurately. Furthermore, it can perfectly detect a video without being limited to image formats. I have finally designed a user-friendly and easy-to-use UI that makes it easier for users to interact with my project.

CHAPTER 5: DISCUSSION AND EVALUATION

In this section, I will provide a synthesis of the accomplishments attained throughout the course of this project. Building upon the findings presented earlier, a comprehensive discussion and evaluation of these results will ensue. This evaluation encompasses a thorough examination of the entire thesis, beginning with the methodology. The focal point of this discussion will be the in-depth exploration and assessment of the YOLOv8 algorithm, analyzing both the level of comprehension of the algorithm and the efficacy of methodological alterations. Next, I will evaluate the implementation of this thesis in relation to the methods and processes that I laid out earlier. From there you will see the effectiveness of the process I have proposed.

Subsequently, attention will shift to the appraisal of the AI training outcomes, incorporating an analysis of the statistical tables and charts derived post the AI training process. The evaluation will extend to the discernment of AI's Object detection ability—it's capacity to recognize objects consistently, irrespective of external conditions or random variables. Following this, a scrutiny of the User Interface (UI) will be conducted, emphasizing aspects such as user-friendliness, ease of navigation, and aesthetic embellishments.

In conclusion, this chapter will culminate with a concise summary encapsulating the key points discussed and evaluated within its contents.

5.1 Evaluation

My methodology starts with designing the structure of the thesis. My design helped project implementation become extremely smooth. Start with learning and researching knowledge related to training AI and related technologies. This has provided a solid foundation of knowledge so that the implementation process can go smoothly. This is an extremely important step and has been my first priority. From that knowledge, I proceed to design the plan and then implement it. Thanks to the precision from step one, these steps are done very quickly and accurately. Finally, re-evaluating the results gives me the most objective view of my entire project. From there, it can be seen that the structural design of my thesis was successful and brought about very good performance.

Next is research on the model and algorithm of YOLOv8. Understanding the structure and algorithm of YOLOv8 will bring extremely important and important factors in implementing this project. The structure of this model is made with 3 main parts: Head, Neck and Spine. These three parts perform separate but closely related functions; Information streams are processed and transmitted continuously and according to rules. Backbone processes input data, extracting features through convolutional layers. Neck connects these features, optimizing information flow. Head generates predictions, object class estimates, and confidence scores for the final result. Then I researched data augmentation and found ways to improve this part. I changed the model so that the input data will add a lot of random elements such as changing brightness, contrast, rotating images, etc. From there, the AI will be trained with a wider level of data and Being less predictable helps greatly increase AI's recognition ability. At the same time, to increase accuracy while evaluating confidence values, I also divided the dataset into 2 parts: train and validate. It can be seen that this change brings great efficiency to the AI training process.

Finally, the most important part is the algorithms of the YOLOv8 model. After researching, I concluded that YOLOv8 determines the data on the image by dividing the image into equal grid cells and begins comparing and assigning reliability data to the cells. this grid and start the comparison loops. Next, I researched and analyzed to understand the algorithms and principles of Compute Losses, Grid Division, Balance Losses, Eliminate Grid Sensitivity, and Build Targets. Mastering these algorithms gives me a detailed and accurate perspective on how the model works.

After training the model, we will have a training folder inside containing all the results including data evaluation chart files. To be able to evaluate the results after training using the above method, let's evaluate it in parallel with the method I used in the pre-thesis.

Confusion matrix

A confusion matrix is a tabular representation used in classification to evaluate the performance of a machine-learning model. In which the vertical axis represents the Predicted and the horizontal axis represents the True label. When comparing the two matrices, we can completely see that the method I used for the Thesis has helped the model to correctly distinguish almost all items and backgrounds from each other. While there is still an error of 25-50% when predicting items together. The matrix of the premise parts still has great confusion in the background with other products and cannot be calculated yet.

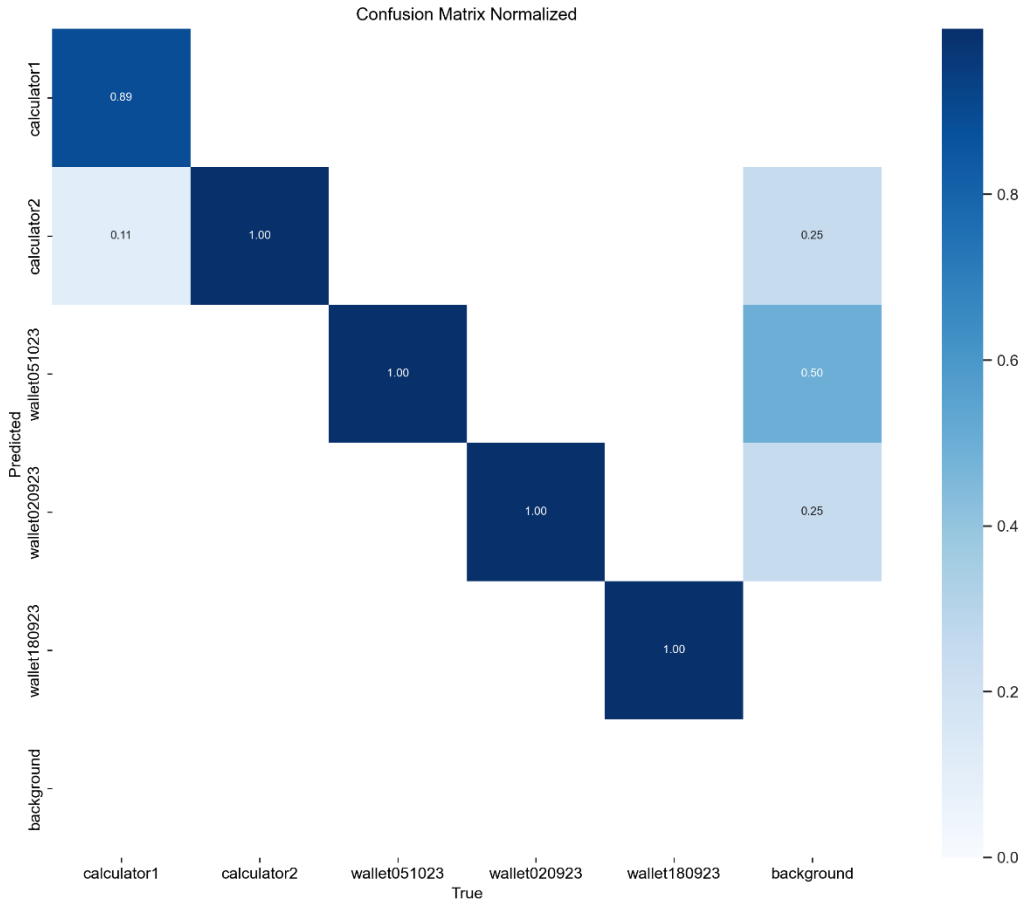

Figure 33 Confusion Matrix Nomalized of Thesis

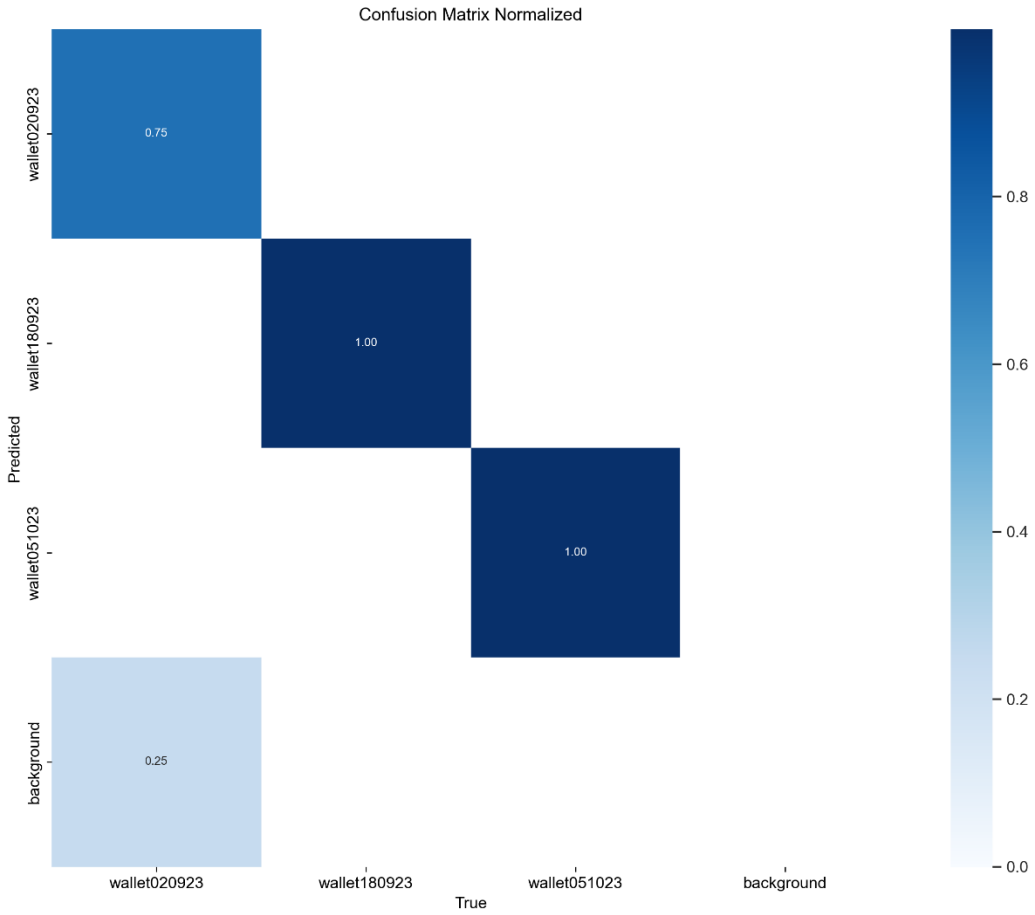

Figure 34 Confusion Matrix Nomalized of Pre-thesis

F1-Confidence curve

The F1-Confidence curve chart shows the correlation between two values, the F1 score and the Confidence value of each subject. F1 is a value that evaluates performance in object detection tasks, and the Confidence value represents the model's confidence in its predictions. When comparing the two tables together, we can see that the thesis will achieve the highest overall F1 score of 0.96 with a very high confidence level of 0.64, while in the pre-thesis the highest overall F1 score is only 0.94 and very high reliability. just below the average level of 0.435. This shows that the model's accuracy has improved significantly.

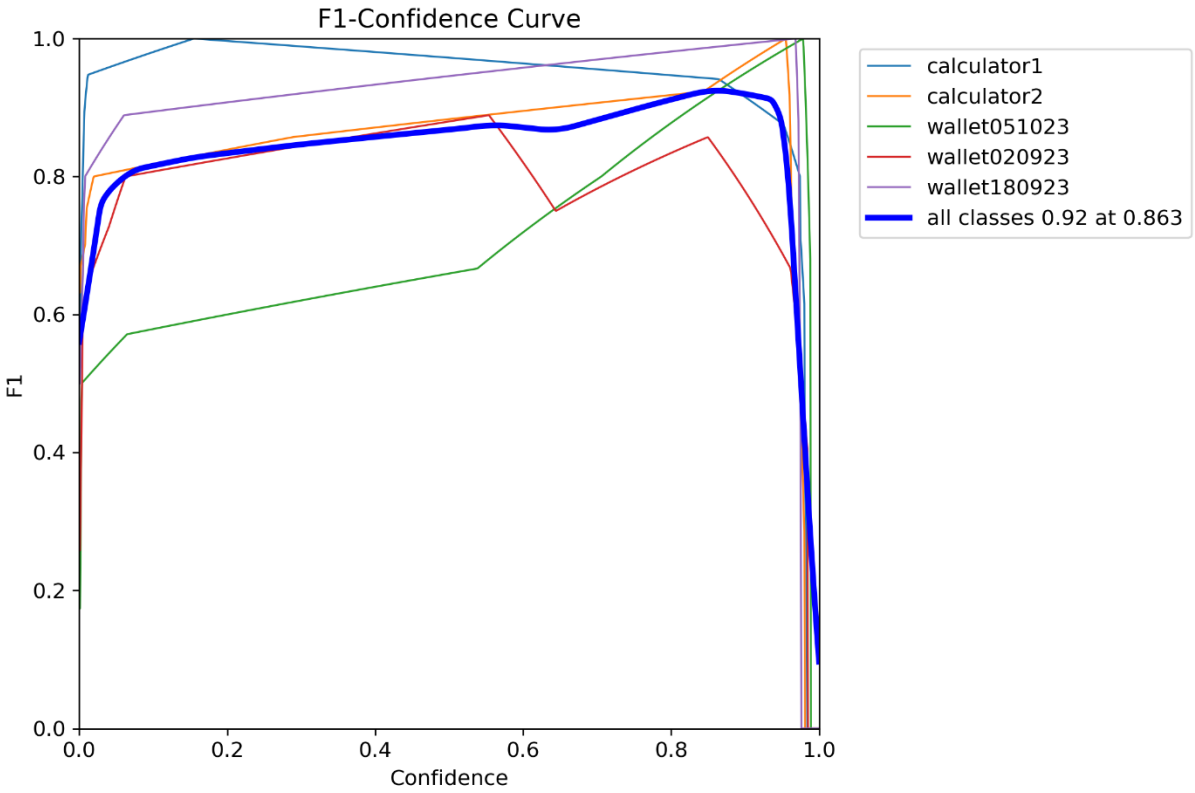

Figure 35 F1-Confidence Curve of Thesis

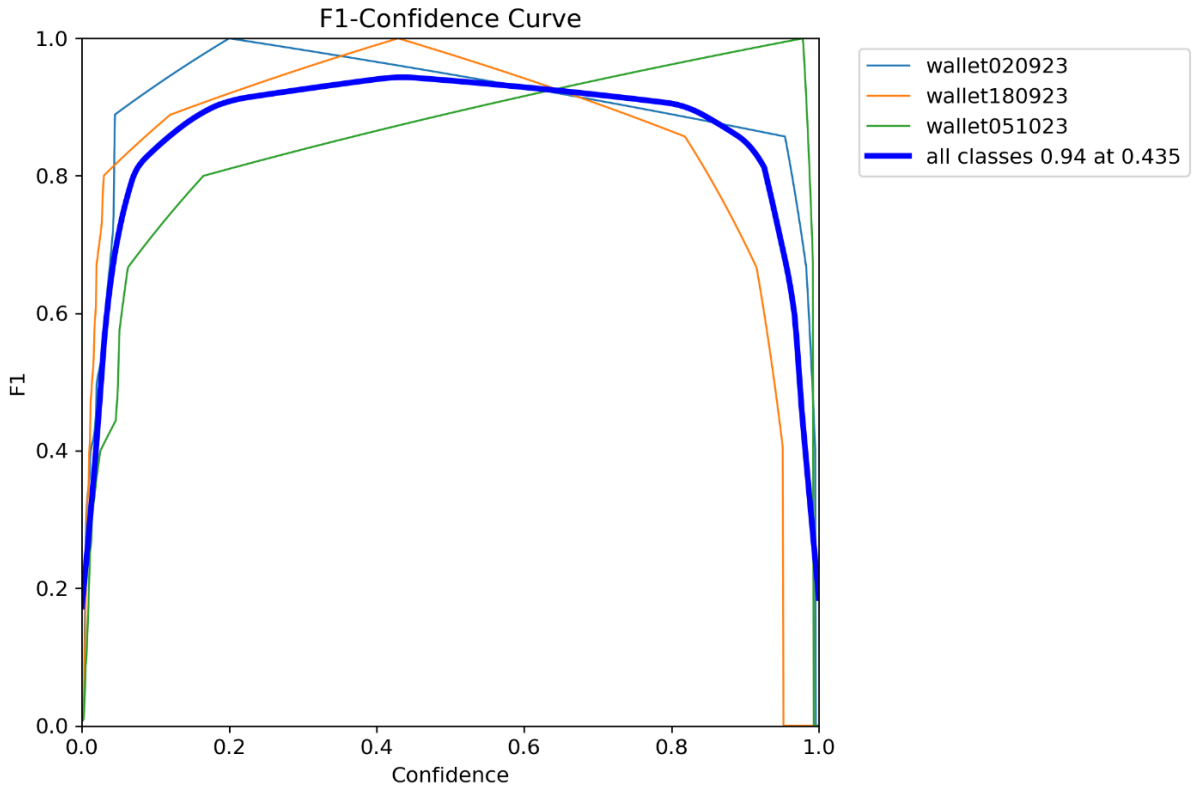

Figure 36 F1-Confidence Curve of Pre-thesis

Accuracy

To evaluate the accuracy of an AI model for object detection, you often use units such as Mean Average Precision (mAP) or Intersection over Union (IoU). Here I will choose mAP as the main measure of my model's accuracy. mAP is calculated by averaging the Average Precision (AP) across classes. mAP helps measure a model's ability to detect objects correctly across many different types of objects. At the same time, it will also be based on two main scales: mAP50 and mAP50-95.

mAP50: This is the mean Average Precision calculated at an IoU threshold of 0.5. It measures how well the predicted bounding boxes overlap with the ground truth bounding boxes when using a relatively lenient IoU criterion.

mAP50-95: This represents the mean Average Precision progressively over multiple IoU thresholds, typically from 0.5 to 0.95 with a step size (e.g., 0.05). It provides a more comprehensive evaluation of the model's performance across a range of IoU thresholds, capturing both lenient and stricter matching criteria.

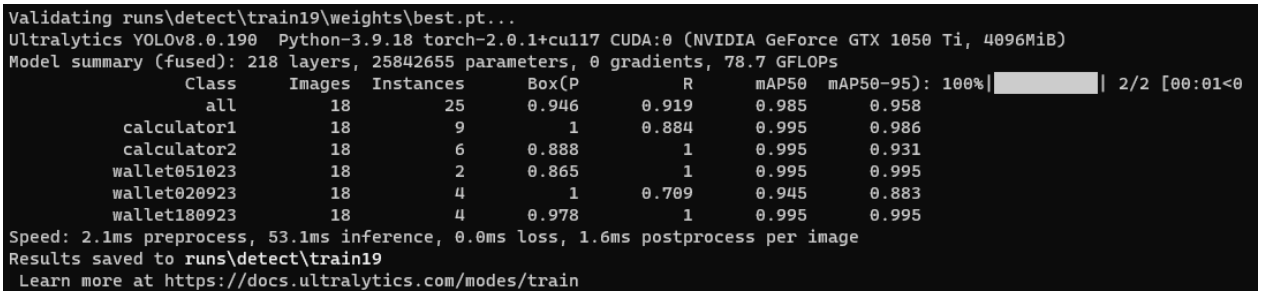

Figure 37 Validation set accuracy

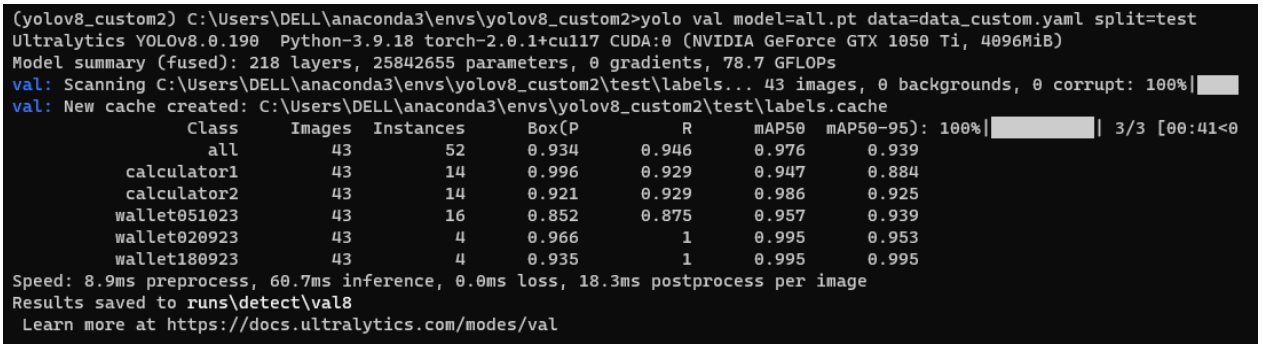

Figure 38 Test set accuracy

From the 2 images above we can see that the average mAP50 index in all labels is 0.985 on average in the validation set, and 0.976 in the test set. The average mAP50-95 index across all labels is 0.958 in the validation set, and 0.939 in the test set. From there we can evaluate the accuracy of my AI is very high, very accurate. This further confirms that my AI training method brings very high and stable performance.

Other Values

In addition, in the statistical data table that changes the sum of other values, the model in the thesis also shows stability and achieves better results. For example, with loss data such as box_loss, cls_loss,... the final result is always at the lowest level while also maintaining its stability. While in pre-thesis, the data changes strongly and continuously and the output results still show a higher number of errors. At the same time, the metrics thesis values also show stability and better final results.

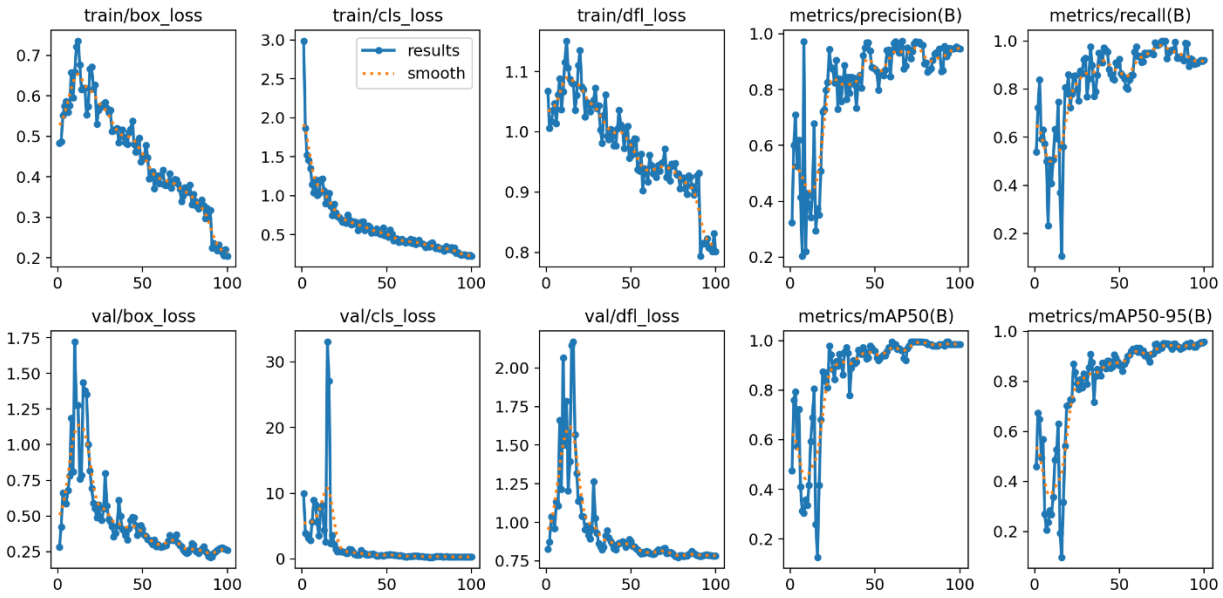

Figure 39 Others values graph of Thesis

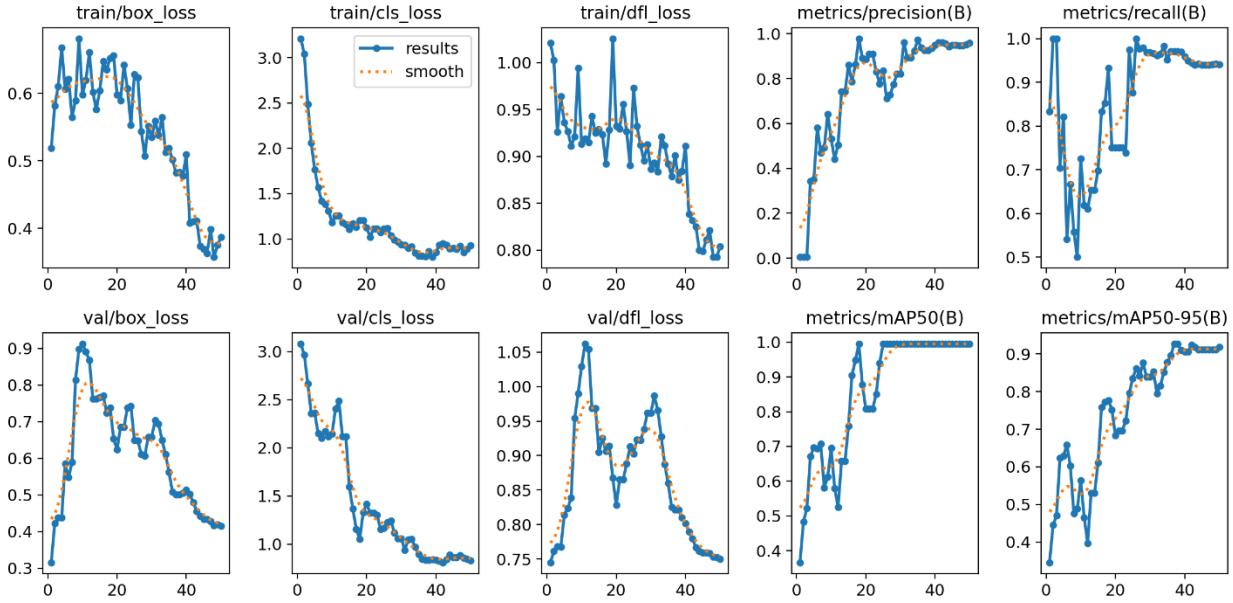

Figure 40 Others values graph of Pre-thesis

For the deployment part I used Anaconda as a support platform to create and control the environment for this project. After researching and learning, I came to the conclusion that using Anaconda is extremely necessary and greatly supports throughout my implementation process. Next, I researched the platforms used to annotate images and among them I decided to use LabelImg as the main application for annotating. Thanks to its easy-to-use and not too heavy interface, this application can be used by even non-professionals and is also very easy to

download and install. Next, to improve the accuracy and reliability of the data throughout the training process, I separated the input training image into 2 parts: training and validation. This greatly contributes to the significantly increased accuracy of the model when detected after training. In addition, thanks to research, testing and statistics, I have derived a certain formula to train the model with values such as epoch, batch, worker... Combined with parallel testing and research. However, with 5 different YOLOv8 model configurations (n, s, m, l, x), I have found that YOLOv8 provides the best performance with the current hardware conditions. Since then, AI training has become more accurate, saves hardware memory, RAM, and saves many other resources such as time, etc. Finally, I have designed the UI for the system. system. The UI is designed and decorated to be eye-catching but still easy to see and user-friendly. This helps users not need to be knowledgeable about information technology but can still use AI to find their items in the easiest way.

The results achieved in Detect Objects were mentioned in section 4.5, so here we will only review the results. In terms of accuracy, my model is almost 98% accurate. The model can identify items of the same type, shape and color that are almost the same without any confusion. Furthermore, the model will not confuse items that have never been trained with items that have been trained. This is a huge step forward compared to what was achieved in the Pre-thesis. At the same time, the accuracy and real-time detection ability of the model is once again confirmed through the ability to detect throughout a video and is no longer tied to the image file. Through research and experimentation, the model's accuracy in detecting objects is almost perfect and accurate on each frame.

5.2 Overall

In this chapter, I summarized the achievements of my project, discussing the YOLOv8 algorithm's implementation and the effectiveness of my methodology. I evaluate the AI training outcomes, emphasizing confusion matrices, F1-Confidence curves, and other statistical values. The analysis reveals improved accuracy and stability compared to pre-thesis results. Additionally, I detail the implementation process, from utilizing Anaconda to annotating images with LabelImg. The UI design is highlighted for its user-friendly features. Lastly, I evaluate object detection, noting the model's impressive 98% accuracy, even in real-time video detection. This chapter concludes with a concise summary of key discussions and evaluations.

CHAPTER 6: CONCLUSION AND FUTURE WORK

Finally in this section we will review all that has been achieved in this project. We'll go over the knowledge I've gained while researching YOLOv8, the best AI training process, and the practical results I've achieved. From the core things that have been done, I will look forward to the next steps in the future of this project. Improvements in algorithms, models, adding new features to the app, ...

6.1 Conclusion in Research

After researching the YOLOv8 model, I realized the capabilities of the Backbone-Neck-Head structure. It coordinates smoothly in transmitting and processing data in each part but is still linked together, making the model operate extremely smoothly and accurately. Next, we know that the detection of this model is dividing the image into grid cells and assigning confidence values. Along with that are the following algorithms to calculate, measure and produce results after training. Furthermore, after testing, measuring and statistics, I have compiled a table of data about the versions of YOLOv8 model and concluded that the YOLOv8m version is the most optimal in terms of performance and hardware.

After implementing the project, I have concluded the AI training process as follows. The first is to select dataset. To train the model for each object, the instructor is required to curate a minimum of 25 categorized images. This process involves capturing 18 distinct photos of the object against various backgrounds and lighting conditions. Subsequently, the collected images are partitioned into two sets: 15 images contribute to the Training set for model training, while 3 images are allocated to the Validation set for performance comparison. Likewise, for the remaining 7 images of the object, they are photographed alongside other objects listed for classification and joint training. Among these, 2 images are designated for the Validation set, and the remaining 5 images are incorporated into the Training set for comprehensive model learning. Next, we need to calibrate and annotate the data to complete the input data for training. Then use anaconda to help set up a virtual environment for the project. And finally, after researching and training many times, I have concluded that the most optimal formula for training an object is: yolo task=detect mode=train epochs=80 data=data_custom.yaml model=yolov8m. pt imgsz=640 batch=8 patience=128 workers=1

In addition to the above research, I have also designed an extremely user-friendly interface. The interface is designed with many colors, each color corresponding to a specific function and has detailed instructions at each step to make it extremely easy for users to use. At the same time, thanks to improvements in AI training, the model's ability to detect items has been greatly improved. My AI can recognize many items that have almost the same shape and color. Moreover, the AI will not be confused with objects that have a similar shape but have not been trained. At the same time, the time when training data has also been optimized from the above research, thereby helping to save hardware resources.

6.2 Future Work

Finally, in this section I will discuss the future development plans of the project. Based on current technologies and predictions about the development of user needs and future technology, I have proposed the following development possibilities for the project in the future:

- Update the next YOLO model version. In the near future, according to information, the next YOLO model is being researched and will be released in the near future. Therefore, once this version is released, updating it to improve more features for this project is necessary.

- Improve the algorithm to save as much data space as possible. Because if this project is put into practice, there will be many items added each year, so data optimization is very urgent.

- Change the algorithm to improve the accuracy and detection ability of the model. Although performance has been significantly improved, further development is still needed.

-Programming apps for Android and IOS so that all IU students can download and use in the simplest and easiest way

-The UI design is more eye-catching and updated with many new features that are difficult to use, order delivery of lost items, and message directly with the student affairs department.

REFERENCES

1. Redmon, J., & Farhadi, A. (2018). YOLOv3: An Incremental Improvement. arXiv preprint arXiv:1804.02767.

2. Wang, A., & Zhang, A. (2020). YOLOv4: Optimal Speed and Accuracy of Object Detection. arXiv preprint arXiv:2004.10934.

3. PyTorch Documentation. (https://pytorch.org/docs/stable/index.html)

4. YOLOv8 Official Repository on GitHub. (https://github.com/ultralytics/yolov3)

5. PyQt5 Documentation. (https://www.riverbankcomputing.com/static/Docs/PyQt5/)

6. Anaconda Documentation. (https://docs.anaconda.com/)

7. Gonzalez, R. C., Woods, R. E., & Eddins, S. L. (2009). Digital Image Processing Using MATLAB. Gatesmark Publishing.

8. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., ... & Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. International Journal of Computer Vision, 115(3), 211-252.

9. Lin, T. Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., ... & Zitnick, C. L. (2014). Microsoft COCO: Common objects in context. In European conference on computer vision (pp. 740-755). Springer.

10. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., & Reed, S. (2016). SSD: Single Shot MultiBox Detector. In European conference on computer vision (pp. 21-37). Springer.

11. He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017). Mask R-CNN. In Proceedings of the IEEE international conference on computer vision (pp. 2961-2969).

12. Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., & Fei-Fei, L. (2014). Large-scale video classification with convolutional neural networks. In Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (pp. 1725-1732).

13. Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

14. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2818-2826).

15. Goodfellow, I., Bengio, Y., Courville, A., & Bengio, Y. (2016). Deep learning (Vol. 1). MIT press Cambridge.

16. Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

17. Everingham, M., Van Gool, L., Williams, C. K., Winn, J., & Zisserman, A. (2010). The PASCAL Visual Object Classes (VOC) Challenge. International Journal of Computer Vision, 88(2), 303-338.

18. Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 4700-4708).

19. Hu, J., Shen, L., & Sun, G. (2018). Squeeze-and-Excitation Networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 7132-7141).

20. YOLOv8 GitHub repository issues and discussions. (https://github.com/ultralytics/yolov3/issues)