Mini-Project Report On

MediTime (Medicine Reminder and Assurance App using Pill Recognition)

Submitted in partial fulfillment of the requirements for the award of the degree of

Bachelor of Technology

in

Computer Science & Engineering

By

Gayathri Ravi (U2003083)
Heynes Joy (U2003095)
Jeffin Jitto (U2003100)
Jocelyn Joshy (U2003103)

Under the guidance of
Dr. Varghese S Chooralil

Department of Computer Science & Engineering
Rajagiri School of Engineering and Technology (Autonomous)
Rajagiri Valley, Kakkanad, Kochi, 682039

July 2023

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
RAJAGIRI SCHOOL OF ENGINEERING AND TECHNOLOGY
(AUTONOMOUS)
RAJAGIRI VALLEY, KAKKANAD, KOCHI, 682039

CERTIFICATE

This is to certify that the mini-project report entitled ***"MediTime (Medicine Reminder and Assurance App using Pill Recognition)"*** *is a bonafide work done by* ***Ms. Gayathri Ravi (U2003083), Mr. Heynes Joy (U2003095), Mr. Jeffin Jitto (U2003100), Ms. Jocelyn Joshy (U2003103)****, submitted to the University in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology (B. Tech.) in Computer Science and Engineering during the academic year 2022-2023.*

Dr. Preetha K. G.	**Mr. Uday Babu P.**	**Dr. Varghese S Chooralil**
Head of Department	Mini-Project Coordinator	Mini-Project Guide
Dept. of CSE	Asst. Professor	Asst. Professor
RSET	Dept. of CSE	Dept. of CSE
	RSET	RSET

ACKNOWLEDGEMENTS

We wish to express our sincere gratitude towards **Dr. P. S. Sreejith**, Principal of RSET, and **Dr. Preetha K. G.**, Head of Department of Computer Science and Engineering for providing us with the opportunity to undertake our mini-project, "MediTime".

We are highly indebted to our mini-project coordinators, **Mr. Uday Baby P.** , Assistant Professor, Department of Computer Science and Engineering, **Ms. Tripti C** , Assistant Professor, Department of Computer Science and Engineering for their valuable support.

It is indeed our pleasure and a moment of satisfaction for us to express our sincere gratitude to our mini-project guide **Dr. Varghese S Chooralil**, for his patience and all the priceless advice and wisdom he has shared with us.

Last but not the least, we would like to express our sincere gratitude towards all other teachers and friends for their continuous support and constructive ideas.

Gayathri Ravi

Heynes Joy

Jeffin Jitto

Jocelyn Joshy

ABSTRACT

It is said 'Health is wealth' and our project aims to increase the quality of life of people running behind wealth by taking care of their day to day health, by reminding our users,both old and young, to have their daily medicines and also verifying whether they had the medicine or not and creating a log based on it.This log will help the doctor in giving the next dosage of medicines as well as the guardians or the loved ones or the housekeeper to take care of them. We also aim to ensure that the user had the medicine for daily usage by keeping a count on the amount of medicine he has in total and how much the user has daily, once the amount is found to be lacking the nearest pharmacy will be contacted immediately for the refill of medicine.We ensure that the user have the medicine by asking the user to capture the image after only which the alarm can be stopped. The number of pills or medicine the user had will be counted and deducted accordingly.This project contributes greatly to the society by giving further business to pharmacies, helping the doctors to make right decisions and increasing quality of life as a whole.

Contents

List of Figures

Chapter 1

Introduction

1.1 Background

1.1.1 Elderly People

The widespread adoption of Smart Medicine Reminders coincides with the significant rise in the number of elderly individuals worldwide. Drug reminder devices, also known as medical alert devices, utilize alarms to prompt individuals or their caregivers to take medication. These advanced systems are essential in our modern society, as they help prevent severe health problems that can arise from forgetting medication or taking incorrect dosages. Recent studies on smart medicine reminders have identified medication delivery to outpatients as the process with the highest error rate in healthcare today. Medication adherence, which refers to how well patients follow medical instructions, typically ranges between 50 and 80 Mistakes in drug administration often occur when patients purchase medications from different pharmacies and use them at home without proper instructions. Many elderly individuals who require medication for their illnesses benefit from proposed smart systems designed to assist them in taking their prescribed medications. Ensuring medication adherence is crucial for reducing mortality rates and overall healthcare costs associated with serious illnesses. However, finding solutions to these persistent healthcare challenges has proven difficult. The use of mobile phones in promoting medication adherence is becoming increasingly necessary as advancements in information and communication technology continue. It is evident from numerous cases that taking the correct medication, in the right dosage, at the appropriate time is crucial. Failing to do so can lead to a range of harmful consequences, from minor health issues to fatality. These risks are amplified in the elderly population, as aging often brings deteriorating capacities such as vision, memory, and logical reasoning, making it challenging for them

to remember which medication to take and when. Additionally, visual impairments and the resemblance of various pills can contribute to confusion and medication errors.

1.1.2 Cognitive disabilities

Cognitive disability is a term applied when an individual has some specific limitations in his/her mental functions and abilities (like social skills, learning, self-help, communication, etc.). The cognitive disability is also called intellectual disability. Cognitive disabilities can be found in millions of people, including people with traumatic brain injury, people with intellectual and developmental disabilities, and people with learning limitations like autism and dyslexia. People with cognitive cognitive impairment may have impaired motor, social, or learning skills that may decrease their performance in the workplace. The cognitive impairments may cause the following restrictions:

- Memory loss for short- or long-term.
- Difficulty with speaking, writing, and reading.
- Problems with orientation.
- Diminished attention period.
- Disability to express oneself, like finding the precise words in a conversation.
- Disability to resolve problems.

Cognitive disabilities range from less serious disabilities (like attention deficit and dyslexia disorder) to more serious disabilities (like hereditary diseases and brain damage).

1.2 Existing System

Smartwatches and Wearable Devices: . They can deliver vibration alerts or notifications to remind users to take their medication. Text Messaging and Phone Call Reminders: Simple but effective, scheduled text messages or phone calls can serve as reminders to take medication. Voice Assistants: Users can set up recurring alarms or ask their voice assistant to remind them at specific times.

1.3 Problem Statement

Elderly people living alone often forget to have their medicines on time and sometimes even skip their medicines. An Application that reminds and assures that they take medicines on time and is therefore a need.

1.4 Objectives

- **Pill recognition**- To recognize the pills that the patient has to take.
- **Medicine reminder** - To remind and ensure that the elderly people take their medicines on time.
- **Medicine supply** -To ensure regular supply of medicines.
- **Routine** -To ensure monthly check ups go smoothly.

1.5 Scope

To assist individuals in improving their medication adherence, our product aims to address the common issue of forgetfulness, particularly among the elderly. We offer an app that simplifies the process and serves as a medicine reminder. The effectiveness of our product can be determined by analyzing the health report, given by doctor , and also the feedback of the respective user or guardian who takes care of the patient.

Chapter 2

Literature Review

Numerous studies have highlighted the prevalence of medication non-adherence among elderly individuals. Factors such as forgetfulness, complex medication regimens, cognitive decline, and lack of caregiver support contribute to this issue. The literature underscores the importance of innovative solutions to enhance medication adherence in the elderly, as poor adherence can lead to adverse health outcomes and increased healthcare costs.The use of mobile health applications, commonly referred to as mHealth interventions, has gained prominence in recent years. Researchers have explored the potential of mobile apps to improve medication adherence among different population groups, including the elderly. Studies demonstrate that mHealth interventions can significantly enhance medication adherence rates and promote better health outcomes. Caregiver involvement plays a crucial role in ensuring proper medication adherence. Studies have highlighted the positive impact of caregiver support and engagement in the medication management process. Mobile applications like MediTime facilitate seamless communication between elderly users and their caregivers, fostering a sense of collaboration and accountability in medication routines.

2.1 Shortcomings of existing medicine reminder apps

1. Limited Customization: Many current medicine reminder apps offer limited customization options, making it challenging for users with complex medication regimens or specific dosing instructions to effectively set up reminders.

2. Lack of Flexibility: Some apps have fixed reminder schedules, which may not cater to users' changing routines or varying dosing frequencies, leading to missed doses.

3. No Caregiver Involvement: Many apps lack features that allow communication and

involvement of caregivers or family members, making it difficult for elderly or dependent users to receive assistance in managing their medications.

4. Ineffective Authentication: Some apps do not utilize advanced technologies like pill recognition models (e.g., YOLO) to verify medication intake, leaving room for uncertainties in tracking actual compliance.

5. Complex User Interfaces: Overly complex interfaces can be daunting for elderly or technologically inexperienced users,leading to confusion and potential non-adherence to the app.

6. Limited Pharmacy Integration: Lack of integration with pharmacies can make it difficult for users to order medication refills directly from the app, leading to potential delays in receiving essential medications.

7. Lack of Tracking and Reporting Features: Some apps lack comprehensive tracking and reporting features, preventing users from monitoring their medication adherence over time or sharing relevant data with healthcare providers.

8. Offline Functionality: Several apps heavily rely on internet connectivity, posing challenges to users in areas with poor network coverage or limited data access.

2.2 Pill Recognition

The application of computer vision and deep learning techniques in pill recognition models has shown promising results in healthcare applications. Researchers have investigated the use of Convolutional Neural Networks (CNNs) and specifically the YOLO (You Only Look Once) model for pill recognition. The YOLO model's real-time object detection capabilities have demonstrated high accuracy in identifying pills from images, making it a suitable candidate for applications like MediTime. The Pill recognition system implemented in MediTime harnesses the power of the YOLO (You Only Look Once) model, a state-of-the-art deep learning algorithm, to accurately identify and authenticate medications from images. This advanced technology has revolutionized the way pill recognition is performed, offering numerous benefits and advantages for applications like MediTime.

2.3 YOLO Model

The YOLO model operates on the principles of convolutional neural networks (CNNs) and object detection. YOLOv5 incorporates SiLU and Sigmoid activation functions. SiLU, also known as Sigmoid Linear Unit or swish activation function, is applied to the convolution operations in the hidden layers. On the other hand, the Sigmoid activation function is used for the convolution operations in the output layer. The outputs of YOLOv5 consist of the detected objects' classes, bounding boxes, and objectness scores. To compute the classes loss and objectness loss, the model employs BCE (Binary Cross Entropy). Additionally, the location loss is computed using CIoU (Complete Intersection over Union) loss.

2.3.1 Working

1. Input Image: The YOLO model takes an input image captured by the user through the MediTime app's camera.

2. Feature Extraction: The image undergoes a series of convolutional layers, extracting hierarchical features to understand its visual content.

3. Bounding Box Prediction: The model predicts bounding boxes that enclose the pills present in the image. Each bounding box consists of coordinates that define the position and size of the pill.

4. Class Prediction: YOLO simultaneously predicts the class of each bounding box.

5. Authentication and Notification: Once the pill is identified, the MediTime app generates a notification confirming the medication intake. This notification is then sent to the designated guardian, along with a timestamp for accurate record-keeping.

2.3.2 Benefits

- Real-Time Object Detection: YOLO excels in real-time object detection, allowing the rapid identification of pills within a fraction of a second. This speed and efficiency are crucial in a medication management application like MediTime, where timely verification of pill intake is essential.

- High Accuracy: The YOLO model has demonstrated remarkable accuracy in object detection tasks, surpassing many traditional algorithms. Its ability to precisely recognize pill shapes, colors, and markings ensures reliable authentication of medication intake, reducing the risk of errors.

- Single Pass Processing: Unlike conventional algorithms that require multiple passes over the image, YOLO performs detection in a single pass, making it significantly faster and computationally efficient.

- Anchor Boxes: YOLO employs anchor boxes to improve the accuracy of object localization. Anchor boxes are predefined bounding boxes of different sizes and aspect ratios, aiding the model in accurately predicting object boundaries.

- Versatility and Generalization: The YOLO model's versatility enables it to recognize a wide range of pill shapes and colors, making it adaptable to various medication types. Its generalization capabilities make it robust against variations in lighting conditions and different camera angles.

The YOLO (You Only Look Once) model has found diverse applications across various industries, owing to its remarkable real-time object detection capabilities and computational efficiency. In the field of computer vision, YOLO has become an indispensable tool for object detection in images, enabling tasks such as surveillance, image analysis, and content moderation.In autonomous vehicles, YOLO plays a pivotal role in enhancing safety and situational awareness by accurately detecting pedestrians, vehicles, and obstacles in real-time. Its speed and accuracy make it ideal for video surveillance systems, empowering security and monitoring operations with swift and accurate object identification. Moreover, YOLO's impact extends into the healthcare industry, where it facilitates medical image analysis, assisting in the detection of abnormalities in medical scans and contributing to disease diagnosis. As the YOLO model continues to evolve and innovate, its versatility and effectiveness are poised to revolutionize various domains, driving advancements in computer vision and artificial intelligence.

2.4 Comparison

Most of the medicine reminder apps offer limited options for customization, which may not cater to the diverse medication needs of users. Users may have complex medication schedules or require specific dosage instructions that cannot be adequately accommodated by rigid reminder settings. This is taken care by MediTime as the app allows the alarm to be set for each and every medicine separately so as to allow flexible medicine intake schedule. Several medicine reminder apps lack features that allow communication and involvement of caregivers or family members. This omission can be particularly problematic for elderly users who may require assistance or supervision in managing their medications. While some apps have basic reminder functionalities, they may not incorporate advanced technologies like pill recognition models (e.g., YOLO) to verify medication intake. The absence of such authentication mechanisms can lead to uncertainties in tracking actual compliance.Some apps lack integration with pharmacies, making it challenging for users to order medication refills directly from the app. This limitation may disrupt the continuity of care, leading to delays in receiving essential medications.

Chapter 3

System Analysis

3.1 Expected System Requirements

The system of user which is a smart phone is expected to have the following features:

- Android platform with a version above 12.
- A storage space of approximate 200 MB for the app.
- A minimum Ram size of 4GB is required in the device.

3.2 Feasibility Analysis

3.2.1 Technical Feasibility

The project is technically feasible since majority of the population are in possession of smartphones. The app only requires minimum requirements to run on a smartphone .

3.2.2 Operational Feasibility

The operations are built in a simple and easy to use manner for elderly people and their care takers. Installation of the app is the only prerequisite operation to be done.

3.2.3 Economic Feasibility

The app can reduce the overhead of expense incurred by elderly people in order to maintain physical assets essential for them to interact with society. The development of the app is also zero budget as it was built using free resources.

3.3 Hardware Requirements

The following are the system requirements to develop the MediTime App.

- Processor: Intel Core i5
- Hard Disk: Minimum 100GB
- RAM: Minimum 8GB

3.4 Software Requirements

The following are the software used in the development of the app.

Operating System: Windows or Linux

3.4.1 Android Studio for flutter app development

Android Studio is a popular Integrated Development Environment (IDE) for developing Flutter apps, as it provides a wide range of tools and features that can help you build high-quality apps faster. Some of the key features of Android Studio for Flutter development include:

A rich set of tools for debugging, testing, and profiling your app. A powerful code editor with support for code completion, refactoring, and more A flexible build system with support for building, testing, and deploying your app. Integration with popular version control systems like Git. A visual layout editor for building attractive user interfaces

3.4.2 Jupyter Notebook

Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations, and narrative text. It is commonly used for data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.

In a Jupyter notebook, you can write and execute code, as well as add text and images to create a rich document that combines code, output, and documentation. You can use Jupyter notebooks for a wide range of tasks, including data visualization, machine learning, and scientific computing.

3.4.3 PyTorch

PyTorch is an open-source machine learning library for Python that allows you to easily build, train, and deploy deep learning models. It is designed to be flexible, efficient, and easy to use, and it offers a wide range of features and capabilities that can be used for a variety of tasks, including computer vision, natural language processing, and scientific computing.

PyTorch provides a variety of tools and features that make it easy to build, train, and deploy deep learning models, including:

A powerful tensor library that allows you to perform operations on multi-dimensional arrays. A flexible neural network library that allows you to define and train complex models Utilities for loading and preprocessing data. Integration with popular optimization algorithms and loss functions. You can use PyTorch to build and train a wide range of machine learning models, including convolutional neural networks (CNNs), recurrent neural networks (RNNs), and long short-term memory (LSTM) networks. PyTorch is widely used in research and industry, and it is a popular choice for building deep learning models.

Chapter 4

Methodology

4.1 Proposed Method

- We plan to build an application that serves as an alarm in order to remind elderly people to take their medicine on time.
- Guardian can enter the details of medicine and set alarm to start ringing at a particular interval.
- Application also contains features like medicine entry and deletion, alarm service, camera to capture image followed by pill recognition, notification send to guardian to ensure the intake of medicine, ordering new supply of medicine.
- Develop a mobile application that can recognize pills using YOLO model.

4.2 MediTime

The primary objective of MediTime is to enhance medication management for elderly users, ensuring they receive timely reminders and adhere to their prescribed dosages accurately. The app aims to provide peace of mind to guardians or caregivers by enabling them to actively participate in the medication process and facilitate timely replenishment of medications through seamless interactions with the pharmacy.MediTime comprises three distinct interfaces: the User Interface, the Guardian Interface, and the Pharmacy Interface. Each interface serves a crucial role in facilitating effective medication management and promote a coordinated approach to elderly healthcare.

The User Interface forms the core component of MediTime, catering to the elderly users' medication needs. This interface offers an intuitive and user-friendly platform that allows users to access their medication details, receive timely alarms for medication

intake and ensure capturing of the medicine taken by the user.The cutting-edge YOLO model, integrated into MediTime's pill recognition system, instantly processes the image, accurately identifying the pill in real-time. The authenticated intake triggers a notification to the guardian, complete with a timestamp, thus creating a transparent and accountable record of medicine consumption.

The Guardian Interface empowers caregivers or family members to play a proactive role in the medication process and will allow them to order a new batch of medicine.They also receive real-time notifications whenever the user successfully verifies their medication intake through the YOLO model. This transparent communication fosters trust and reassurance in the caregiving process. To maintain a seamless supply chain for medications, MediTime integrates a Pharmacy Interface that allows guardians to request replenishment of essential medicines when they are running low. This interface facilitates smooth communication with the pharmacy, ensuring timely delivery of the required medication to the user's specified location.

4.3 Pill Recognition

Central to MediTime's efficacy is the implementation of the YOLO (You Only Look Once) model for pill recognition. The pivotal pill recognition system, ensures authenticity and accountability in medication intake. This futuristic technology enables users to verify their medication intake by capturing an image of their pills through the app's camera. The model accurately analyzes the image, validating the consumed medication and notifying guardians of the successful intake.Developed with a vision to enhance medication management for elderly users, YOLO serves as the backbone of MediTime's seamless and accountable approach to verifying medication intake.

4.4 YOLO Model

YOLO v5 is the 5th generation of YOLO. It is famous for its detection accuracy and prediction speed.. YOLO v5 adds Mosaic data augmentation method to the training pictures. Through random scaling, random cutting, and random layout, four different pictures are mixed into one picture. In batch normalization calculation, the data of four pictures are calculated at one time, therefore the mini batch size does not need

to be very large, especially suitable for single GPU training. The backbone of YOLO v5 is the combination of focus module and CSP darknet53 structure. Its neck combines Feature Pyramid Networks (FPN) and Path Aggregation Network (PAN). It includes four connection layers, four convolution layers, and five CSP layers. Given the size of input is 640*640*3, by feature partitioning, three outputs with sizes of 20 * 20 * 255, 40 *40* 255 and 80*80*255 are produced. They are used for detection of different sizes.Choosing an activation function is crucial for any deep learning model, for YOLOv5 went with SiLU and Sigmoid activation function.

4.4.1 YOLO Model in MediTime

When it is time for the user to take their prescribed medication, the MediTime app prompts the user with an alarm to capture an image of the medication using their device's camera. Upon capturing the image, the YOLO model springs into action, analyzing the photograph with unrivaled speed and precision.By eliminating the need for complex post-processing, YOLO efficiently recognizes and validates the pill depicted in the image in real-time.Upon successful pill recognition, the YOLO model promptly triggers notifications to the designated guardian or caregiver. This proactive system allows continuous communication between the elderly user and their support network, providing reassurance to guardians that the medication intake has been acknowledged.

Through the integration of the YOLO model, MediTime instills a heightened sense of accountability and compliance in the medication management process. Elderly users are empowered to take charge of their healthcare, while guardians are equipped with the necessary tools to actively monitor and participate in their care recipients' medication routines.As the development of MediTime progresses, the YOLO model will continually evolve to encompass a wider array of medication types, ensuring its relevance and effectiveness across various healthcare scenarios. Future enhancements may include refining recognition capabilities, improving processing speed, and integrating advancements in the field of deep learning to further elevate MediTime's pill recognition system.

Chapter 5

System Design

5.1 Architecture Diagram

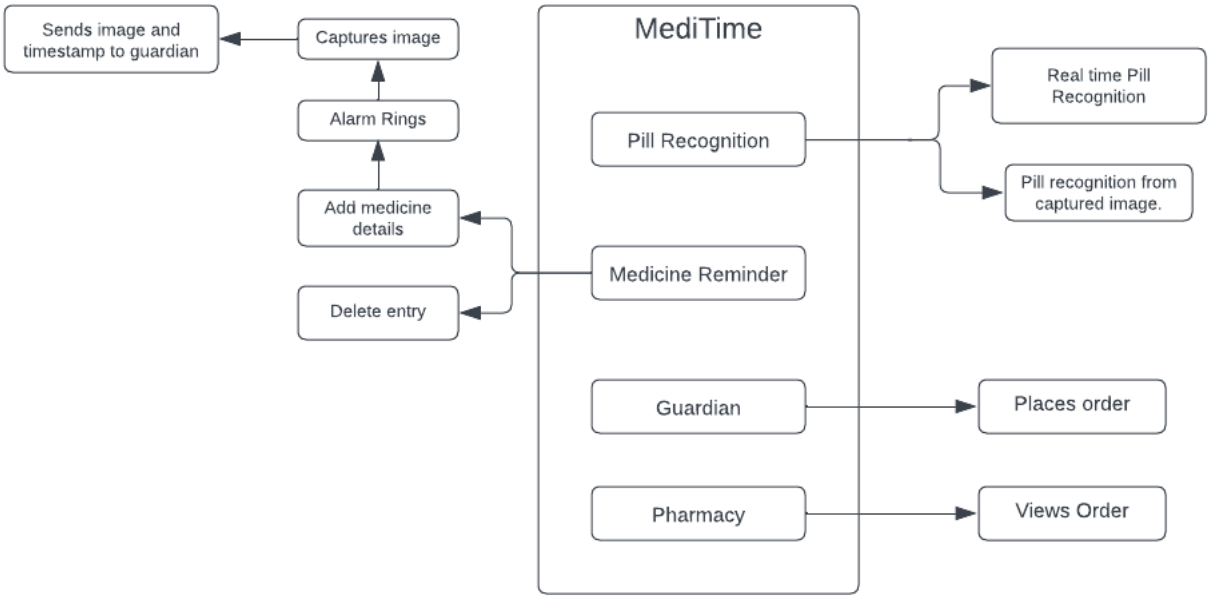

Figure 5.1: Architecture diagram

5.2 Sequence diagram

5.2.1 Pill Recognition

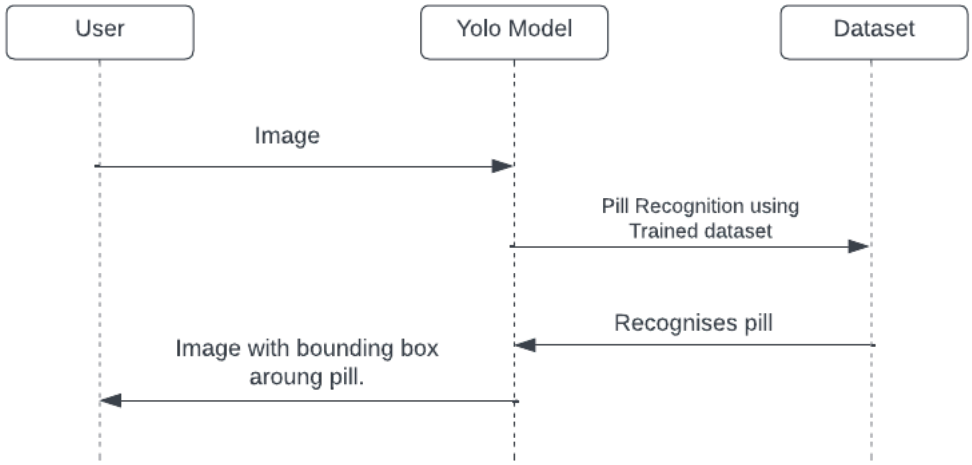

Figure 5.2: Pill Recognition

5.2.2 Image sending

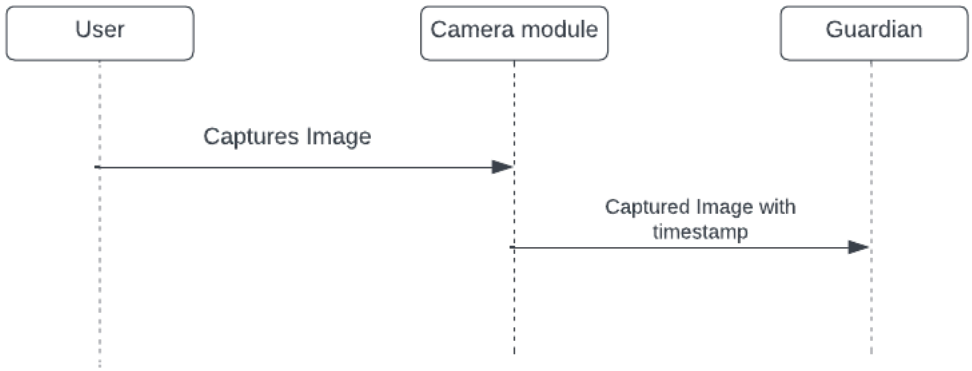

Figure 5.3: Image sending

5.2.3 Medicine Reminder

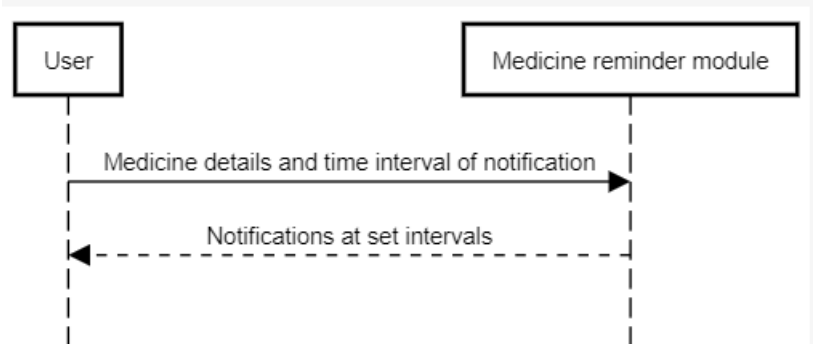

Figure 5.4: Medicine Reminder

5.2.4 Medicine Entry

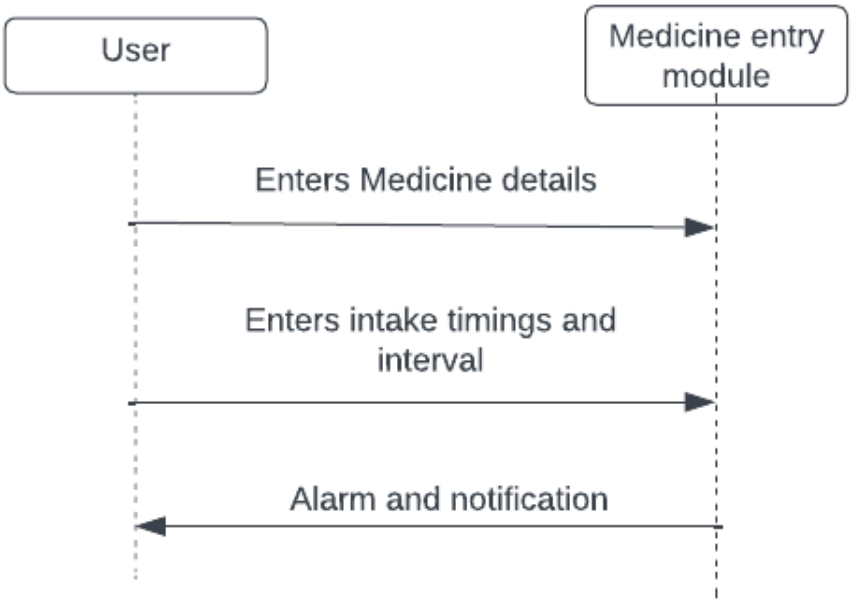

Figure 5.5: Medicine

5.2.5 Entry Deletion

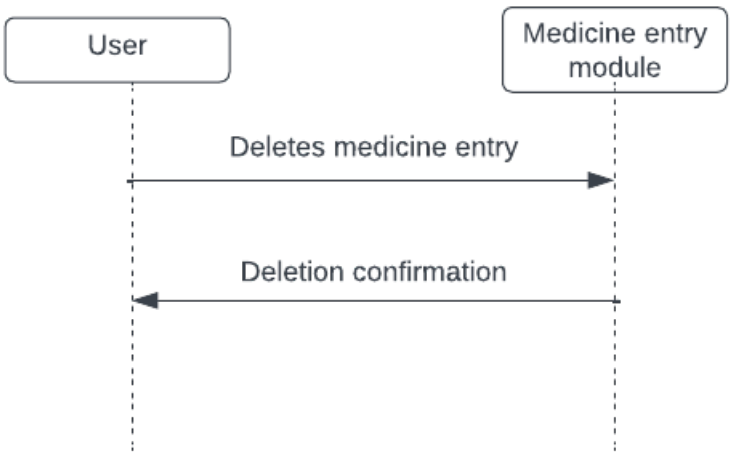

Figure 5.6: Entry Deletion

5.2.6 Guardian

Figure 5.7: Guardian

Chapter 6

System Implementation

6.1 Pill Recognition

6.1.1 Model Architecture

YOLO v5 is the 5th generation of YOLO. It is famous for its detection accuracy and prediction speed.YOLO v5 has a simple network structure, consisting of input, backbone, neck and prediction.

Input:YOLO v5 adds Mosaic data augmentation method to the training pictures. Through random scaling, random cutting, and random layout, four different pictures are mixed into one picture. By these means, the background information of training image is enriched, which is very beneficial to small target detection.Additionally, when calculating batch normalization, the data of four pictures are calculated at one time, therefore the mini batch size does not need to be very large, especially suitable for single GPU training.

Backbone: The backbone of YOLO v5 is the combination of focus module and CSP darknet53 structure. Focus module slices one data information into four, and then enate them on channel dimension.This module is designed for reducing FLOPS and increasing speed, rather than mAP increase. CSPDarknet53 contains 29 convolutional layers, and a 725×725 receptive field. Its ability of feature fusion is much better than original Darknet53.

Neck: Its neck combines Feature Pyramid Networks (FPN) and Path Aggregation Network (PAN). It includes four connection layers, four convolution layers, and five CSP layers. It can speed up the transmission of feature information and feature fusion.

PANet is a feature pyramid network, it has been used in previous version of YOLO (YOLOv4) to improve information flow and to help in the proper localization of pixels in the task of mask prediction. In YOLOv5 this network has been modified by applying the

CSPNet strategy.

SPP block performs an aggregation of the information that receives from the inputs and returns a fixed length output. Thus it has the advantage of significantly increasing the receptive field and segregating the most relevant context features without lowering the speed of the network.

Prediction: Given the size of input is 640*640*3, by feature partitioning, three outputs with sizes of 20 * 20 * 255, 40 *40* 255 and 80*80*255 are produced. They are used for detection of different sizes.

Choosing an activation function is crucial for any deep learning model, for YOLOv5 went with SiLU and Sigmoid activation function.SiLU stands for Sigmoid Linear Unit and it is also called the swish activation function. It has been used with the convolution operations in the hidden layers. While the Sigmoid activation function has been used with the convolution operations in the output layer.

YOLOv5 returns three outputs: the classes of the detected objects, their bounding boxes and the objectness scores. Thus, it uses BCE (Binary Cross Entropy) to compute the classes loss and the objectness loss. While CIoU (Complete Intersection over Union) loss to compute the location loss.

6.1.2 Dataset

The Roboflow dataset consisting of 451 distinct pill images, is used for training the YOLO model used above. Roboflow dataset consists of 351 celebrity images for training, and 98 of them in the testing, each of them are distinct. The training set is 39GB. From that, a 2GB test set was used to train the last dense layer. For training and testing, the input images have to go through the same pre-processing that is defined by the Roboflow.

6.2 Medicine Reminder

When the user logs in from the login page, they get sent to the medicine reminder page. The page contains a display that shows the number of reminders currently set and a button to add a new reminder.

Once the button to add a new reminder is pressed, it opens up a page where the user can add the name of the medicine, its dosage, the type of medicine (bottle, pill, syringe,

etc..), the interval at which reminder should work and the starting time of the reminder.

Once a reminder has been made, the user also gets the option to delete that reminder.

6.3 Alarm and Notification

Upon the user setting the time of reminder and the interval at which the alarm should ring, eg: 6,12,24 hours, the application uses the smartphone's speaker and the phone's display over all apps feature to ring the alarm and display a pop out notification which prompts the user to take a picture of the medicine and send it to the guardian to turn the aarm off and disable the notification.

6.4 Order List

When the quantity of medicine nears zero,the guardian of the patient has the feature to place orders of the next stock of medicine. Using SQLite we were able to store the data in a database which the guardian can input and submit using the 'place order button' and the list of multiple orders were saved into an order page which the guardian can navigate to by pressing the 'View Orders' button.

6.5 Sharing and Timestamp

After the application rings the alarm upon the specified time,the app turns the camera on and after detecting the pill,is made to send to the guardian's Whatsapp along with a timestamp of when the picture was taken to assure the guardian that the patient has taken the medication on time and give him/her relief for the sam.

Chapter 7

Testing

7.1 Pill Recognition

Our modified YOLO model was trained on the test set of Roboflow dataset. It was tested using 9% of the dataset and 20% of the dataset is used as a validation set.

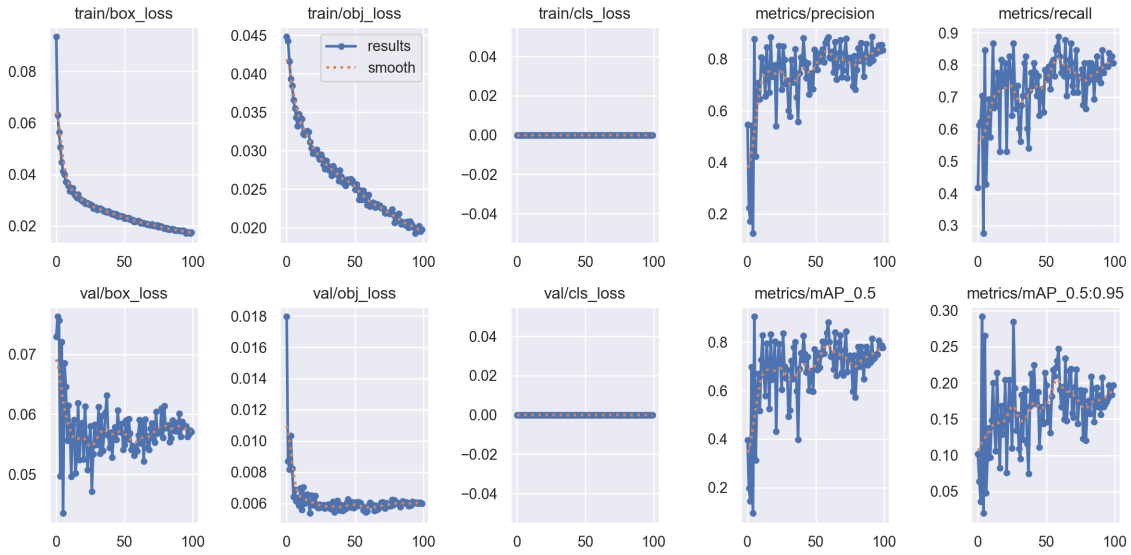

Figure 7.1: Result graph

The Result graph shows how the loss in our modified model shows a constant decrease over 100 epochs with batch size of 8. With each epoch,the bounding boxes get tighter to the ground truth model and the precision increases as the object loss and class loss decreases.

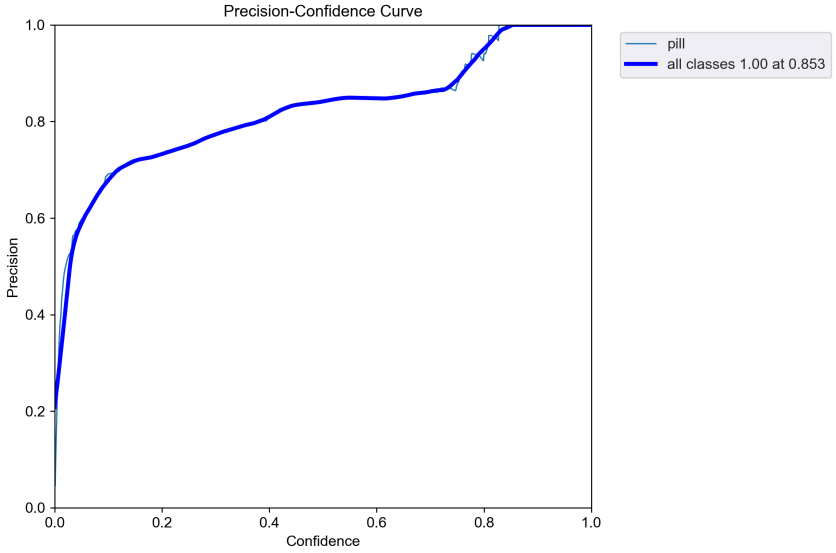

Figure 7.2: Precision-Confidence Curve

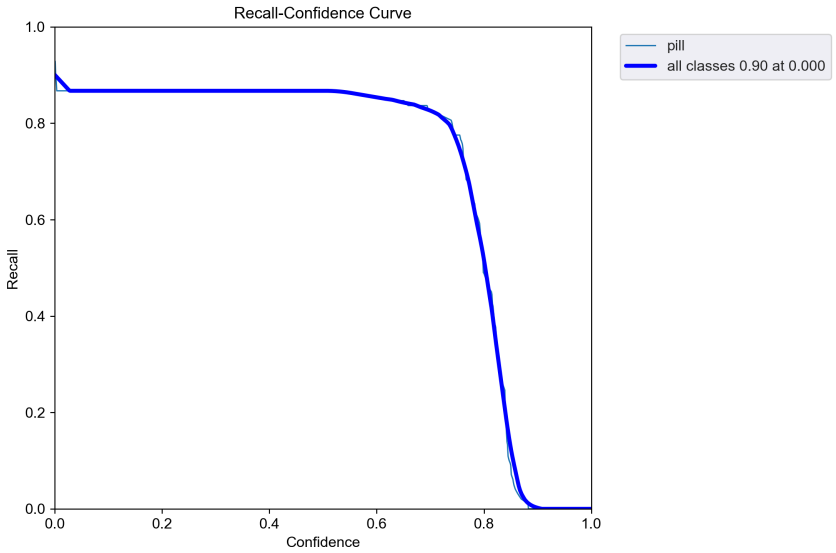

Figure 7.3: Recall-Confidence Curve

From the above 2 graphs which compares the values of prediction and recall with confidence,it shows that the accuracy of prediction of our model is high and the confidence percent of each image that has been predicted is high as well.

Chapter 8

Results

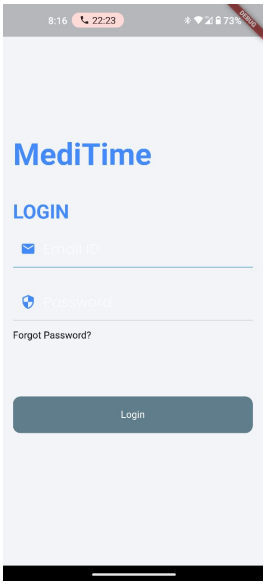

Figure 8.1: Login Page

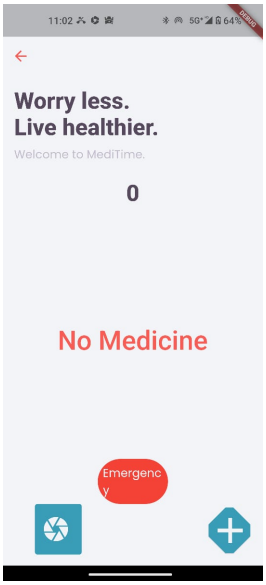

Figure 8.2: Main Screen

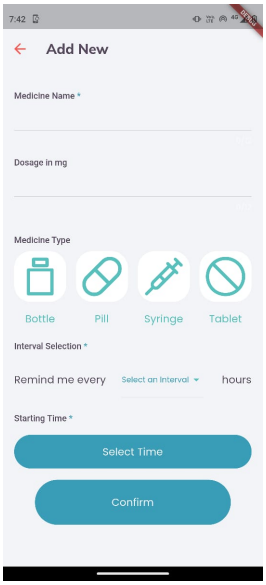

Figure 8.3: Medicine Entry

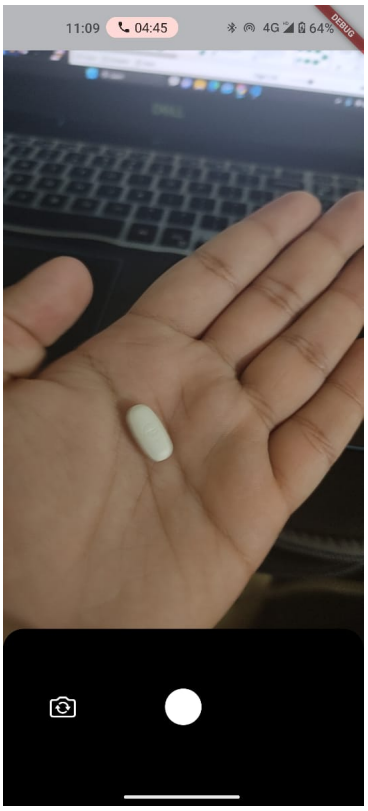

Figure 8.4: Camera Interface

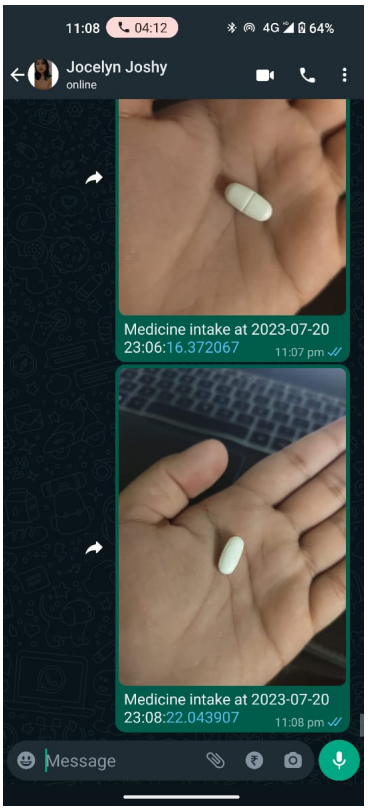

Figure 8.5: Image sending with timestamp

Figure 8.6: Pill Recognition

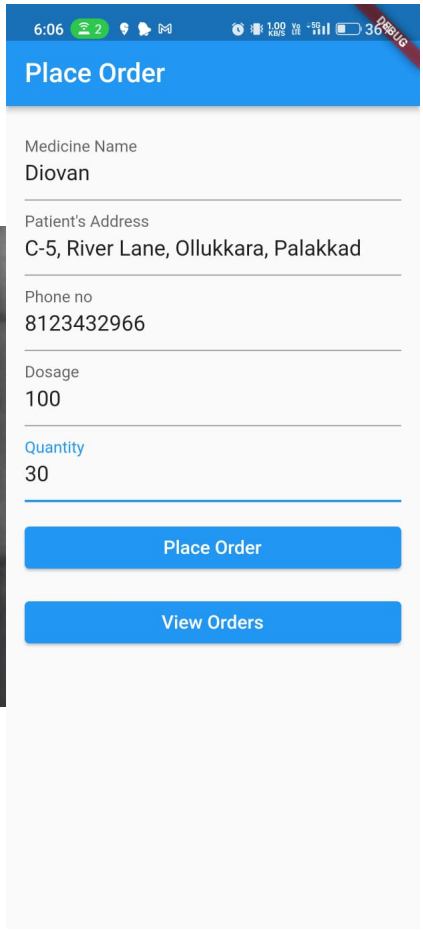

Figure 8.7: Guardian interface

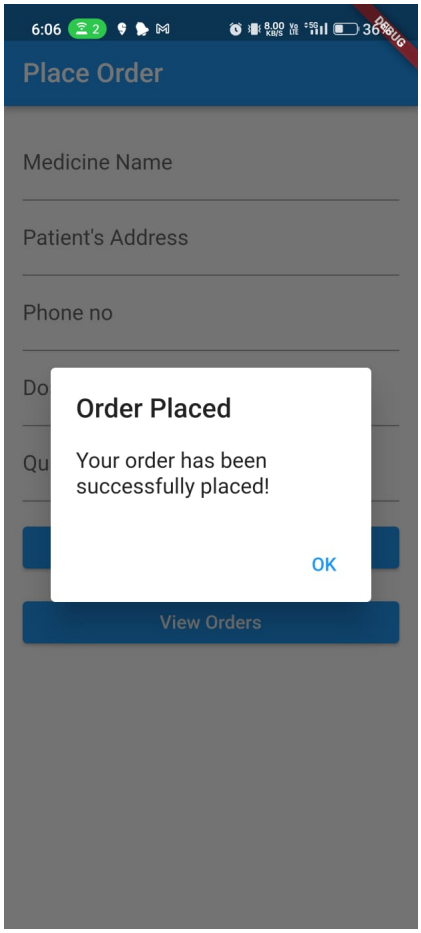

Figure 8.8: Success Screen

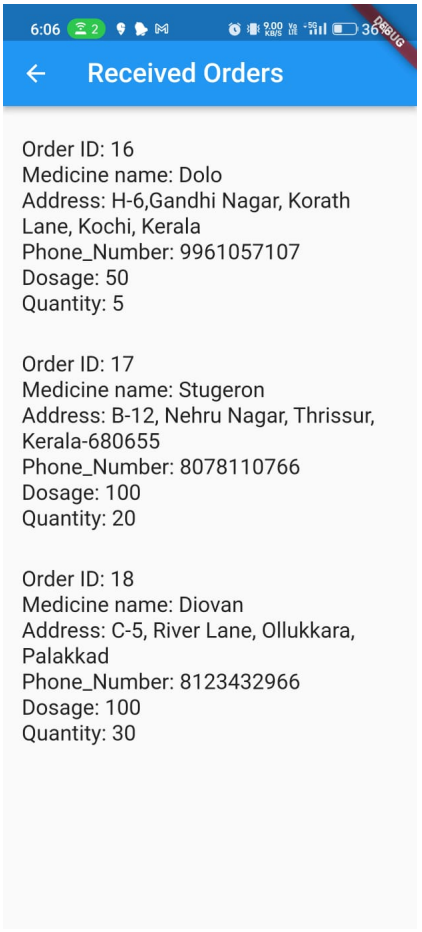

Figure 8.9: Pharmacy interface

Chapter 9

Risks and Challenges

1. Pill Recognition: The captured image should not have a lot of noise.
2. App Familiarization: Patient needs to get familiar with the various interfaces of the app.
3. Device Permissions: Access must be granted on the device for features to run, such as camera for image capturing.
4. Integration of all the Features.
5. Depreciation of existing functions.

Chapter 10

Conclusion

We have developed an android application that addresses the critical challenges of medication management for elderly individuals. Through its user-friendly interfaces, seamless communication between users, guardians, and pharmacies, and the implementation of advanced pill recognition using the YOLO model, MediTime transforms the way elderly users interact with their medications. The app consist of the following features: medicine reminder, entry and deletion of medicines, capturing image, pill recognition, sending image and timestamp to guardian, place order and view order by guardian and pharmacy respectively

We have used YOLO Model to implement pill recognition.The YOLO-powered pill recognition system further enhances caregiver engagement, allowing guardians to actively participate in the medication process.MediTime empowers users to take control of their healthcare, ensuring timely and accurate medication intake. With MediTime, the elderly can confidently embrace a healthier and more independent lifestyle, while caregivers and guardians find reassurance in a well-coordinated and secure healthcare management system.

References

[1] M.-Y. Wang, P. H. Tsai, J. W. S. Liu, and J. K. Zao, "Wedjat: A Mobile Phone Based Medicine In-take Reminder and Monitor," in 2009 Ninth IEEE International Conference on Bioinformatics and BioEngineering, Jun. 2009, pp. 423–430. doi: 10.1109/BIBE.2009.60.

[2] C. Crema, A. Depari, A. Flammini, M. Lavarini, E. Sisinni, and A. Vezzoli, "A smartphone-enhanced pill-dispenser providing patient identification and intake recognition," in 2015 IEEE International Symposium on Medical Measurements and Applications (MeMeA) Proceedings, May 2015, pp. 484–489. doi: 10.1109/MeMeA.2015.7145252.

[3] E. J. MacLaughlin, C. L. Raehl, A. K. Treadway, T. L. Sterling, D. P. Zoller, and C. A. Bond, "Assessing Medication Adherence in the Elderly," Drugs Aging, vol. 22, no. 3, pp. 231–255, Mar. 2005, doi: 10.2165/00002512-200522030-00005.

[4] Girshick R, Donahue J, Darrell T, et al. Rich feature hierarchies for accurate object detection and semantic segmentation[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2014: 580-587.

[5] Pang J, Chen K, Shi J, et al. Libra R-CNN: Towards Balanced Learning for Object Detection[C]// 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, 2020.

[6] Schroff, F., Kalenichenko, D. and Philbin, J., 2015. Facenet: A unified embedding for face recognition and clustering. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 815-823).

[7] Redmon J, Divvala S, Girshick R, et al. You Only Look Once: Unified, Real-Time Object Detection[J]. Computer Vision & Pattern Recognition, 2016.

Appendix A: Base Papers

A Smart Medicine Reminder to help patients: A Review

SA Sathsara
Department Of Computer Science
General Sir John Kotelawala Defense University
Ratmalana, Sri Lanka
37-cs-5969@kdu.ac.lk

Dr. Wijendra Gunathilake
HOD – Department Of Computer Science
Senior Lecturer Grade II
General Sir John Kotelawala Defense University
Ratmalana, Sri Lanka
wijendrapg@kdu.ac.lk

***Abstract*—Smart Medicine Reminders are playing a major role in elderly people's lives. With the increase in age, most human beings are having many kinds of different incapabilities in their life cycles. In that case, taking medicine at the correct time with the correct dosages and identifying the correct medicine according to their sicknesses are a must if those elderly people need to avoid severe repercussions. Also, with the busy schedules of the patients and their caretakers, lack of vision and hearing, and lack of memorization of the elderly people, most likely forget to take their medications. Because of these problems, there were some smart mobile applications and smart devices were proposed by some researchers. Those are alarm systems, voice-enabled systems, wristbands, pill dispenser boxes, etc.**

Keywords— Smart Mobile Applications, Smart Devices, Sensors, Smart Medicine Dispensers

I. INTRODUCTION

The rise of Smart Medicine Reminders has been accompanied by the sharp increase in the prevalence of the population growth of elderly people all over the world. What exactly are drug reminder devices? A medical alert device is a gadget that utilizes an alarm to remind people to take their medicine on their own or with their caretakers. In this modern world, we are living today, we need the help of these smart systems specially to get rid of unnecessary severe health problems. Forgetting to take medicine at the correct times and taking medicine according to the wrong dosages can be caused patients to have problematic severe situations [5]. According to recent research on smart medicine reminding [1], the delivery of medications to outpatients was shown to be the process with the highest mistake rate in contemporary healthcare. Medication adherence, defined as "the amount that the patient follows medical instructions," has been observed to range between 50 and 80% [6]. Patients who purchased prescription and over-the-counter medications from different pharmacies and utilized them at residence without sufficient instruction caused many drug administration mistakes. In most family backgrounds some elderly people are having sicknesses and are supposed to get medicines. There are some proposed smart systems that are helping the patients like that to take medicines for their sicknesses.[2] Medication adherence is clinically important in lowering serious illness mortality and overall costs for health care. Nonetheless, the aforementioned health issues have long resisted resolution [7]. Because mobile phones are simple for patients to carry even in the case of an emergency or disaster, a system employing mobile phones to promote medication-taking will become more required as part of the mHealth (mobile health) system as information and communication technology advances [8]. After witnessing so many of these situations, it is clear that taking the proper tablet by the correct person at the exact time is critical; otherwise, taking the wrong pill or not having one at all may confront the patient to a variety of harmful circumstances ranging from minor health difficulties to death [9]. These facts are easily explained when we realize that once a person reaches old age, many of their capacities, such as field of vision, memory, and logical capacities, tend to deteriorate proportionately to age, making it challenging for them to remember which pill to take when, recalling taking them, or confusing one pill with another as a result of their decreased vision and the similarity of the pills[10].

II. LITERATURE REVIEW

In Ref. [03], A mobile phone-based medicine in-take reminder and monitor called Wedjat has been made. Their patient medication administration has been identified as the most non-error procedure in nowadays healthcare. Under or overdoses due to erratic in-takes, interactions between drugs and foods, and un-reconciled drug prescriptions are some of the most common reasons for medication errors. In this paper, the mobile application which is called "Wedjat" is designed to assist patients to take their medicine correctly, taking those medications on time, and storing the previous drug takings if any case is needed. There are three basic functions in this application. They are, providing medication reminders, providing medication identification and dosage instructions, and keeping a record of medicine that patients had taken within the dosages and times.

Especially this wedjat app has two most important features. Firstly, it can alert the patients before taking their medications, and the next one is revising the drug takings when patients couldn't able to take their medicine. This mobile computing application combines real-time scheduling algorithms and mobile phone-based telemonitoring approaches to provide ubiquitous services to many out-patients.

In Ref. [04], there is a paper about a secure mobile-enabled assisting device for diabetics monitoring which was named SMEAD. By time, the evolution of the medical sector is getting developed with the use of modern technology such as robots which are used for diagnosis-based procedures, smart mobile applications, and electronic healthcare devices. And thanks to the Internet of Things, biometric sensors can detect patients' vital parameters and send them to a store that is based on cloud service systems for further consultancy and further analysis. And millions of data of the patients can be easily stored in those clouds. In this paper, a healthcare system called SMEAD is an end-to-end secured system that has been made for assisting diabetes patients.

This includes a piece of wearable monitoring equipment to detect ad guess the status of diabetic patients and a MEDIBOX that is produced to provide an alert about the time at which medicine must be taken as a reminder and about the correct dosage of medicines. Insulin must be stored at a cool temperature between 36-46 Fahrenheit. They can be stored in this proposed MEDIBOX. As above mentioned, this has a blockchain-based security system with cryptographic securities and has structured data access for the medical community using smart contracts for so many medical organizations and communities. As a special feature, if there are medical issues like forgetting to take medicine at the correct time or insecure blood sugar levels, this SMEAD system is sending alerts to patients' takers of care via social platforms like Facebook, Twitter, and WhatsApp. Because of their usage of them, the alerting path will be connected continuously so that can help to save patients having fatal health problems.

There are quantitative advantages of linked medical devices, such as a decrease in mortality rates, fewer clinic visits, emergency room visits, and hospitalizations, along with a decrease in hospital bed days of care and duration of stay. Specially this paper has shown that remote monitoring can be more helpful in the medical sector before many numbers of issues that the patients and caretakers have. There are three devices that can be worn, two smart bands that can be worn on the wrist and neck and smart footwear. With the vibration and flex sensors in the neckband, food and water intake can be measured. Vibration can take to analyze the chewing pattern and a flex sensor can take to measure the pressure while swallowing. Also, there are optical heart rating sensors, galvanic skin sensors, and temperature sensors. While monitoring sugar in blood altitude plays the main role while measuring the temperature. Smart footwear can measure the loss or gain of patients' weight. All these gathered data will be sent to the smartphone via the cloud systems which can be forwarded to the superior with the help of the AppIOT platform. Also, the MEDIBOX design is intended to be a user-friendly device that keeps pharmaceuticals such as insulin in a suitable environment and reminds patients to take their prescriptions at the appropriate times. The suggested comprehensive system focuses primarily on these three essential components to construct a comprehensive mobile-based protected healthcare system in an intelligent AppIoT platform that can forecast diabetes conditions in real-time. In an emergency, the doctor can securely access the patient's e-health information and prescribe an appropriate dose.

The delivery of medications to outpatients was shown to be the process with the highest mistake rate in contemporary healthcare. Patients who purchased prescription and over-the-counter medications from different pharmacies and utilized them at residence without sufficient instruction caused many drug administration mistakes. Here the authors are discussing an IoT-based Smart medicine reminder device that will be designed according to the difficulties that have been faced by the elders. All over the world, the population of elderly people is increasing. When people are getting old their mobility ability is getting decreased. So to look after them, the help of modern technology is a must. This technology can be their mango friend when they are alone. So It is very smart to make an IoT-based device to control these problems. Here authors took their objective to develop an IoT-based smart reminder device. There are some special features in the proposed medicine reminder system; a locked pill dispenser that helps to give medicine at specific times, a medical alert system that helps to give a signal to the patients to take their medication, a refill tray that can help to refill the medication when the stock has finished, multiple caregivers, a feature can cause to enable one to care about multiple patients and IoT connection which producing network connections among physical attributes. The delivery of medications to outpatients was shown to be the process with the highest mistake rate in contemporary healthcare. Patients who purchased prescription and over-the-counter medications from different pharmacies and utilized them at residence without sufficient instruction caused many drug administration mistakes. The pill replenishment notices to users on the patient's contact list guard against unintended occurrences of the patient forgetting to resupply their medications. One of the weaknesses of this proposed medical reminder is that this is not suitable for patients who are continuously forgetful. The primary impact of this study is the creation of a prototype that would remind older people to take their medications on time and much other important information [11].

As mentioned above, [11] the use of medicine is the most basic method to prevent of having many kinds of diseases. So it is very essential to take medicine at the correct prescribed time. So, this MED-Alert helps to send reminders to take whatever the medicine is at the correct time by using auditory and visual reminding systems [12]. For the audio alerting, they use a buzzer Arduino system, a small speaker that can be connected directly by an Arduino. And for visual alerting, a LED system is being used. There is a special feature in this MED-Alert that if a patient doesn't take medicine, an email will be sent to his/her family member as a reminder. And a doctor can get a report on the weekly usage of patients' medicine which will be sent via email. Here the main system is built by using a Raspberry Pi 2 B. IoT, or the Internet of Things, is a general term for the email-based alert system.

In Ref. [2], wireless cellphones and personal digital assistants are growing as information hubs connect their human users with various electronic gadgets and the World Wide Web. As a result, they have rapidly become the de facto foundation for customized information services. The Kannon project team at Taiwan's National Chiao Tung University (NCTU) is creating a ubiquitous service infrastructure to assist senior healthcare. Among their deliverables is the Health Pal PDA Phone, which can interact using Wireless devices, universal plug-and-play (UPnP) e-home cloud platforms, and online healthcare providers to give older people 24/7 healthcare services. This article highlights the early outcomes of this endeavor, including Health Pal's functional and operational principles, as well as its design's activity-oriented approach.

Telephones, particularly their wireless descendants, are possibly the most widely used, important, and constantly expanding technology in contemporary history. Visionaries predict that these small, inexpensive, untethered communication/information processing devices will have a plethora of applications, including camcorders, game consoles, navigation aids, tour guides, house keys, remote controls, burglar alarms, health monitors, electronic wallets, passports, and so on. Economists saw them as being the most appropriate methodology for bridging the "digital gap". We regard them as a potential tool to reconnect the elderly with their lives and respective surroundings. To persuade yourself of the potential utility of cell phones in enhancing and protecting the lives of the elderly. Health Pal was created to be the user's "continuous friend." Its design procedure consists of four steps: the production of visual metaphors and symbols, the arrangement of user interface design including clickable buttons and menu, the planning of user-device interactions, and the design of display screens and formats. Health Pal is currently in its early stages. The physical gadget and its user interfaces were designed using the activity-oriented design (AOD) approach. However, hardware prototype and software implementation have only just begun. Once the functioning prototype is complete, its usability will be evaluated further [2].

In ref. [15], the mainly focused thing was the Smart medicine pill reminder box. There was not any smart app to act as a reminder. There are voice functions and display functions to help the patients via the auditory method and visual methods. Also, these functions could be more helpful for differently-abled persons. There are 3 boxes according to the usage of different tablets. If the voice function said to take the paracetamols and suddenly the appropriate LED for that medicine will glow. It will be very easy for adults to take their medicines without any faults. There is an alarm inbuilt with this system which can help patients to give alerts even they are sleeping. The main advantage of this reminder, there is no need of having people around to remind the patients to take their medicine. But according to my point of view, this advantage is risky to adhere to when the patients are in very severe conditions.

By referencing [16], we can see the paper is about autonomous medicine reminding applications in mobile operating systems. It is called MEDiDEN. This has some unique features; categorized medicine packages, a reminder function that can be worked in mobile operating systems, and can be updated with the latest news on the medication field. Here for the classification of medications deep-learning architecture has been used. And for the identification process of medications, they have used the Inception V3 method and the Inception V4 method. By the processed results of these two-identification methods Inception V4 has more accuracy than Inception V3. But the researchers have used Inception V3 because of its ability to speed up the system and due to its smaller size. Majorly python language has been used to develop the backend and to run the Neural network model. Most people get many health issues when they become elderly. With their lack of vision ability, they cannot even realize their medication. So, for that, the researchers have implemented a function to check the medication by taking a photo of them. Also, the patients can be able to set a reminder to take their medications on time. Here we can see that the image identification method is the most unique function in this system. Patients must align their medication package to be fitted to the red square as seen in the mobile application under the image classification. Within a few seconds, the images which have been sent to the server after taking images can be classified through the neural networks. When patients are setting their reminders they can include the dosages, the time that the medicine needed to be taken, and whether to take medicine before or after having meals. In the proposed application, the researchers used main three processes: a collection of datasets, classification methods of medications, and details of implementation and development. This dataset has fourteen medications so far. In dataset collection, it has divided into two sections: the training data set and the test data set. The training set has 52500 photos spread throughout 14 classes with 3750 images in each. Also in the testing set, there are 90 photos in each class for a total of 1260 images.

III. METHODOLOGY

The purpose of this research is to review smart mobile applications and smart reminder devices related to the medication sector which is expanding to remind elderly people to take their medications at the correct time within the correct dosages. And this chapter explains the methodology of the review study, what are the previously proposed systems, and the gatherings of reasons relevant for not taking the medicine.

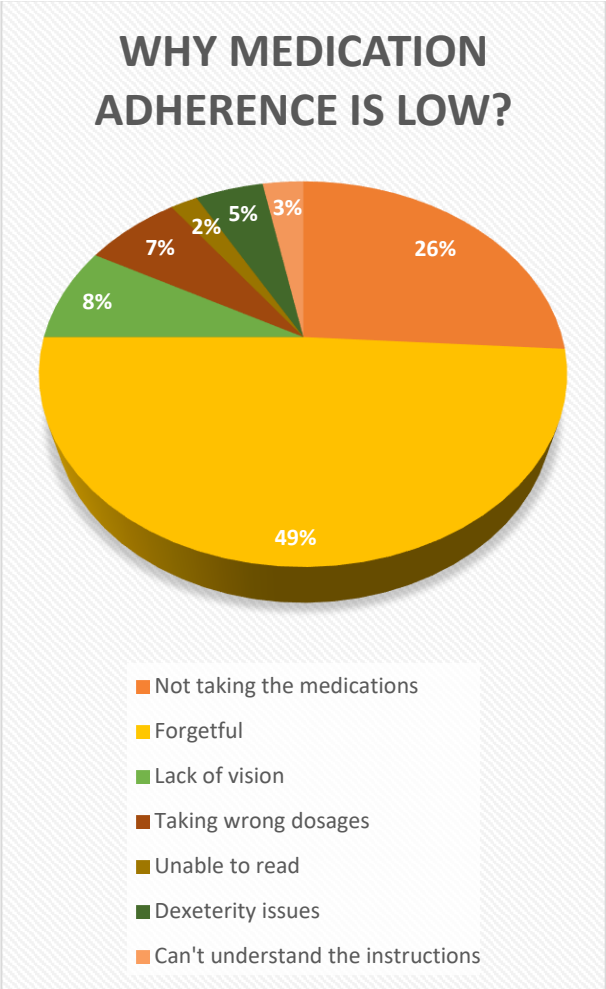

Figure 1

Mainly, some research papers were downloaded from the databases such as IEEE Xplore, Research gate, Springer, Elsevier, ScienceDirect, and google Schooler by using keywords. Then I checked and selected similar topics according to my research topic. Then I read and analyzed the selected research papers. According to the stated research topic, the first step was to select keywords and search research papers using those keywords related to smart medicine, smart pill dispensers, IoT devices for elderly people, and Smart mobile applications. For the selection of databases, In IEEE Xplore database found 669 research papers according to the medicine reminders. Then after filtering them for smart medicine dispensers, it found only 49 papers that are closely related to the research topic. In springer, database found 591 research papers and it found only 23 papers that are closely related to the research topic. The highest percentage of papers were selected from the IEEE Xplore database.

In Ref. [14], we can see some main reasons that are directly caused for our above-mentioned problems seen in Figure 1. This is a quantitative method of finding medical adherence according to a set of elderly people.

Figure 2 explains some reviews of past research projects about smart medication reminders.

S/R	Projects	Methodology	Limitations
1.	Medication Reminder with Medicine Dispenser	Image processing is used to scan the prescription.	Sometimes the Pi camera will malfunction, necessitating a CPU restart.
2	Smart medicine pill box reminder with voice and display for emergency patients	If the voice function said to take the paracetamols and suddenly the appropriate LED for that medicine will glow. It will be very easy for adults to take their medicines without any faults. There is an alarm built with this system which can help patients to give alerts even they are sleeping.	Cannot use by deaf and blind people.
3.	The autonomous pill dispenser	A Bluetooth signal is sent to a device via an Android app. The patient must turn the device such that one pill falls into the cone's tip and is then vibrated out of there.	Most elderly people cannot be able to flip their mobile phones due to a lack of physical ability of their body.
4.	MEDiDen: Automatic Medicine identification Using a Deep Convolutional Neural Network	Has these features. Image classification, Medicine classification, and Medication reminder.	Only fourteen types of medications can be identified.
5.	Wedjat: A Mobile Phone-Based Medicine In-take Reminder and Monitor	It can warn patients about possible drug-drug or drug-food interactions as well as help them arrange an appropriate intake strategy to prevent them. If a dosage is missed, it has the capability to automatically change the ingesting schedule.	Cannot use by deaf and blind people.

Figure 2

IV. CONCLUSION

The use of smartphones and mobile apps has significantly expanded in recent years, therefore creating mobile applications and smart devices for healthcare services can have the intended potential development. Despite the fact that there are many different mobile applications for medication reminders, there must be better all-rounder smart medicine apps and devices to help people take their medications correctly and promote better medication adherence. Taking medicine with proper procedures may cause to dismiss the severity of having a crucial time in the patients' lives.

V. DISCUSSION & FUTURE WORK

Thus, by referring to numerous current proposed systems, prior initiatives, and study papers based on drug dispensers and reminders, as well as considering challenges experienced by disabled individuals, we may evaluate what sections of those studies are lacking. Also, this section describes my own vision of research work. By referring to previous research works, this paper shows some features that most authors have considered when they are improving smart medicine reminding devices. Most of the authors have focused on reminding patients to take medicine at the correct time with the correct dosages. We must see the lagging areas in lowering medication adherence and propose new systems for the sake of human beings.

ACKNOWLEDGMENT

The necessary and valuable instructions and guidance for the paper were given by the supervisor Dr.Wijendra Gunathilake [KDU] the guidance for the study model was given by Dr. LP Kalansooriya [KDU] and Ms. GUI Uwanthika [KDU] by providing insight and expertise that greatly assisted the research, although they may not agree with all of the interpretations/conclusions of this paper.

REFERENCES

[1] "Improved Disabled Mobile Aid Application for Android : Health and Fitness Helper for Disabled People | IEEE Conference Publication | IEEE Xplore." https://ieeexplore.ieee.org/abstract/document/8837069 (accessed Oct. 18, 2022).

[2] J. Zao *et al.*, "Activity-Oriented Design of Health Pal: A Smart Phone for Elders' Healthcare Support," *EURASIP Journal on Wireless Communications and Networking*, vol. 2008, Feb. 2008, doi: 10.1155/2008/582194.

[3] M.-Y. Wang, P. H. Tsai, J. W. S. Liu, and J. K. Zao, "Wedjat: A Mobile Phone Based Medicine In-take Reminder and Monitor," in 2009 Ninth IEEE International Conference on Bioinformatics and BioEngineering, Jun. 2009, pp. 423–430. doi: 10.1109/BIBE.2009.60.

[4] M. Saravanan, R. Shubha, A. M. Marks, and V. Iyer, "SMEAD: A secured mobile enabled assisting device for diabetics monitoring," in 2017 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS), Dec. 2017, pp. 1–6. doi: 10.1109/ANTS.2017.8384099.

[5] M. Hayakawa, Y. Uchimura, K. Omae, K. Waki, H. Fujita, and K. Ohe, "A Smartphone-based Medication Self-management System with Realtime Medication Monitoring," *Appl Clin Inform*, vol. 4, no. 1, pp. 37–52, Jan. 2013, doi: 10.4338/ACI-2012-10-RA-0045.

[6] R. B. Haynes, E. Ackloo, N. Sahota, H. P. McDonald, and X. Yao, "Interventions for enhancing medication adherence," *Cochrane Database Syst Rev*, no. 2, p. CD000011, Apr. 2008, doi: 10.1002/14651858.CD000011.pub3.

[7] S. H. Simpson et al., "A meta-analysis of the association between adherence to drug therapy and mortality," BMJ, vol. 333, no. 7557, p. 15, Jul. 2006, doi: 10.1136/bmj.38875.675486.55.

[8] M. N. K. Boulos, S. Wheeler, C. Tavares, and R. Jones, "How smartphones are changing the face of mobile and participatory healthcare: an overview, with example from eCAALYX," *BioMedical Engineering OnLine*, vol. 10, no. 1, p. 24, Apr. 2011, doi: 10.1186/1475-925X-10-24.

[9] C. Crema, A. Depari, A. Flammini, M. Lavarini, E. Sisinni, and A. Vezzoli, "A smartphone-enhanced pill-dispenser providing patient identification and in-take recognition," in *2015 IEEE International Symposium on Medical Measurements and Applications (MeMeA) Proceedings*, May 2015, pp. 484–489. doi: 10.1109/MeMeA.2015.7145252.

[10] E. J. MacLaughlin, C. L. Raehl, A. K. Treadway, T. L. Sterling, D. P. Zoller, and C. A. Bond, "Assessing Medication Adherence in the Elderly," Drugs Aging, vol. 22, no. 3, pp. 231–255, Mar. 2005, doi: 10.2165/00002512-200522030-00005.

[11] S. B. Kumar, W. W. Goh, and S. Balakrishnan, "Smart Medicine Reminder Device For The Elderly," in 2018 Fourth International Conference on Advances in Computing, Communication & Automation (ICACCA), Oct. 2018, pp. 1–6. doi: 10.1109/ICACCAF.2018.8776734.

[12] S. Jayanth, M. B. Poorvi, and M. P. Sunil, "MED-Alert: An IoT device," in *2016 International Conference on Inventive Computation Technologies (ICICT)*, Aug. 2016, vol. 2, pp. 1–6. doi: 10.1109/INVENTIVE.2016.7824900.

[13] S. Chawla, "The autonomous pill dispenser: Mechanizing the delivery of tablet medication," in *2016 IEEE 7th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*, Oct. 2016, pp. 1–4. doi: 10.1109/UEMCON.2016.7777886.

[14] "FUNDAMENTAL RESEARCH ON MEDICATION REMINDER SYSTEM - Google Search." https://www.google.com/search?q=FUNDAMENTAL+RESEARCH+ON+MEDICATION+REMINDER+SYSTEM&oq=FUNDAMENTAL+RESEARCH+ON+MEDICATION+REMINDER+SYSTEM&aqs=edge.0.69i59j69i60.296j0j4&sourceid=chrome&ie=UTF-8 (accessed Oct. 19, 2022).

[15] V. Sree, K. S. Indrani, and G. Latha, "Smart medicine pill box reminder with voice and display for emergency patients," *Materials Today: Proceedings*, vol. 33, Oct. 2020, doi: 10.1016/j.matpr.2020.08.400.

[16] N. Hnoohom, S. Yuenyong, and P. Chotivatunyu, "MEDiDEN: Automatic Medicine Identification Using a Deep Convolutional Neural Network," in 2018 International Joint Symposium on Artificial Intelligence and Natural Language Processing (iSAI-NLP), Nov. 2018, pp. 1–5. doi: 10.1109/iSAI-NLP.2018.8692824.

AUTHOR BIOGRAPHIES

SA Sathsara is an officer Cadet of Sri Lanka Army and a 3rd year undergraduate at General Sir John Kotelawala Defense University. Following the BSc (Hons) computer science.

Dr. Wijendra Gunathilake is Head of the Department and Senior Lecturer within the Department of Computer Science, Faculty of Computing, General Sir John Kotelawala Defense University. program.

Improved YOLO v5 with balanced feature pyramid and attention module for traffic sign detection

Linfeng Jiang*, *Hui* Liu, *Hong* Zhu, and *Guangjian* Zhang

School of Artificial Intelligence, Liangjiang, Chongqing University of Technology, Chongqing, China

Abstract. With the development of automatic driving technology, traffic sign detection has become a very important task. However, it is a challenging task because of the complex traffic sign scene and the small size of the target. In recent years, a number of convolutional neural network (CNN) based object detection methods have brought great progress to traffic sign detection. Considering the still high false detection rate, as well as the high time overhead and computational overhead, the effect is not satisfactory. Therefore, we employ lightweight network model YOLO v5 (You Only Look Once) as our work foundation. In this paper, we propose an improved YOLO v5 method by using balances feature pyramid structure and global context block to enhance the ability of feature fusion and feature extraction. To verify our proposed method, we have conducted a lot of comparative experiments on the challenging dataset Tsinghua-Tencent-100K (TT100K). The experimental results demonstrate that the mAP@.5 and mAP@.5:0.95 are improved by 1.9% and 2.1%, respectively.

Keywords: Traffic sign detection, Convolutional neural network, Feature fusion.

Introduction

Since the rapid development of automatic driving technology, great changes have taken place in people's daily life. At the same time, a large number of new technological developments are in an urgent demand. Traffic sign detection is one of them. The mission of traffic sign detection is to locate traffic signs from given pictures or videos, and then predict the category information of traffic signs correctly. However, it is still a challenging task due to the complex background, various kinds of shapes, together with the shelter of the trees.

With the evolution of deep learning, many people attempt to use convolutional neural network (CNN) based object detection methods to detect traffic sign, such as YOLO [1] and SSD [2] (Single Shot MultiBox Detector). However, the results they achieved are not

* Corresponding author: linfengjiang@cqut.edu.cn

very satisfactory. Besides the accuracy of detection, the models they proposed are always complex, together with a large number of parameters, which is computationally expensive, and is hard to be embedded into the automatic driving terminal.

To get higher prediction speed with lightweight network model, YOLO v5 is employed as the baseline method, which is one of the best object detection methods with excellent detection accuracy and very low time complexity, especially suitable for intelligent driving. Additionally, for a higher detection precision, we have made some improvement on YOLO v5. The proposed method mainly makes the following contributions: (i) Balance feature pyramid structure is used to improve our model, which can enhance the ability of feature fusion, so that both semantic information and position information of traffic sign in small size are taken into account. (ii) Attention module is also added in our model to help feature extraction.

The experimental results show that the accuracy and recall of our model are obviously improved over the baseline method YOLO v5, mAP@.5 and mAP@.5:0.95 are improved by 1.9% and 2.1%, respectively.

The rest of the paper is organized as follows. Section 2 introduces some related work in traffic sign detection. Section 3 gives the proposed method. Section 4 gives the results of our methods and the comparison with other methods. The last section gives the conclusion.

2 Related work

In this section, we briefly review the methods that can be used for traffic sign detection. The study of traffic sign detection can be divided into: traditional approach and deep learning based method.

2.1 Traditional approach

The detection methods based on traditional methods are mainly based on the physical characteristics of targets, the most common methods are color based detection algorithm and shape based detection algorithm. The shape based methods are always based on Hough Transform. Besides, the color based detection method obtains the color information points from the image, then connects them into regions, and finally obtains the interested region.

2.2 Deep learning based method

Object detection methods based on deep learning can be divided into two categories: one-stage methods and two-stage methods. Classic algorithms like SSD, YOLO, and RetinaNet belong to one-stage methods. Some other methods like Fast R-CNN, Faster R-CNN are the symbols of two-stage methods. Usually, two-stage methods are excellent in accuracy, while one-stage methods are better in speed. In this section, we will briefly introduce some classic object detection algorithm.

YOLO [1] was proposed by R. Joseph et al. in 2015. It is the first one-stage object detection algorithm. One-stage object detection methods do not have the process of classification on the region proposal, but directly regresses the output category. YOLO is well known for its accuracy, together with the extremely speed, which is one of the most commonly used algorithms in industrial circle. The core idea of YOLO is to transform the object detection into a regression problem. It feeds pictures into a neural network, and then outputs the bounding boxes and categories of objects directly. Later, Yolo is continuously optimized and improved. Thus, YOLO v2, v3, v4, v5 were proposed. In particular, YOLO v5 greatly improves the accuracy and reduces the size of the model, which will be

introduced in the next section. SSD [2] is another famous one-stage object detection methods. The main contribution of SSD is using small convolutional filters to predict category information and box offset. SSD is superior to the first version YOLO in both accuracy and speed.

Two-stage method is another technology road map for object detection. Different from one-stage methods, two-stage methods extract the depth features of images through backbone network, and then generate region proposal through RPN network. Finally, it determines the class information through two branches of classification and regression. In 2013, Ross et al. proposed RCNN [3] network. It is one of the earliest object detection methods based on deep learning. It made a breakthrough in object detection, and achieved 58. 5% mAP in Pascal VOC 2007 dataset, while DPM [4] only get a mAP of 34. 3% in the same dataset. To improve RNN, Ross g et al. proposed Fast RCNN [5] in 2015. They improved RCNN by inserting SPP-Net module, and use VGG 16 as its backbone. Thus, it gets a better detection accuracy. Later, Faster RCNN [6] occurs in June, 2015. It is the first end-to-end deep learning based object detection method. It introduces Region Proposal Network (RPN) and breakthrough the speed limit of two-stage methods and make a great improvement in detection results. In the next few years, a lot of object detection methods appear, promoting the development of object detection technology.

3 Proposed method

3.1 Brief introduction to YOLOv5

As introduced in section 2.2, YOLO v5 is the 5th generation of YOLO. It is famous for its detection accuracy and prediction speed. As shown in Figure 1, YOLO v5 has a simple network structure, consisting of input, backbone, neck and prediction.

A) Input: As YOLO v4 did, YOLO v5 adds Mosaic data augmentation method to the training pictures. Through random scaling, random cutting, and random layout, four different pictures are mixed into one picture. By these means, the background information of training image is enriched, which is very beneficial to small target detection. Additionally, when calculating batch normalization, the data of four pictures are calculated at one time, therefore the mini batch size does not need to be very large, especially suitable for single GPU training.

B) Backbone: The backbone of YOLO v5 is the combination of focus module and CSP darknet53 structure. Focus module slices one data information into four, and then enate them on channel dimension. This module is designed for reducing FLOPS and increasing speed, rather than mAP increase. CSPDarknet53 contains 29 convolutional layers, and a 725 × 725 receptive field. Its ability of feature fusion is much better than original Darknet53.

C) Neck: Its neck combines Feature Pyramid Networks (FPN) and Path Aggregation Network (PAN). It includes four connection layers, four convolution layers, and five CSP layers. It can speed up the transmission of feature information and feature fusion.

D) Prediction: Given the size of input is 640*640*3, by feature partitioning, three outputs with sizes of 20 * 20 * 255, 40 *40* 255 and 80*80*255 are produced. They are used for detection of different sizes.

3.2 Improved detection model

YOLO has achieved great results in object detection. However, in order to get more feature information, YOLO v5 uses 8 times, 16 times, and 32 times down samplings to detect

objects of different sizes respectively. Thus, a large numbers of position information has been lost. This makes it difficult to detect small objects.

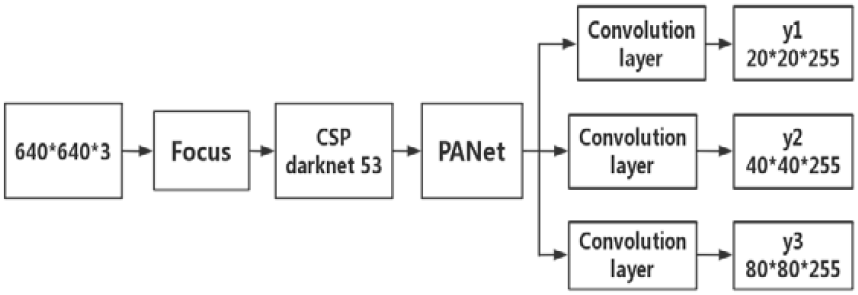

Fig. 1. YOLO v5 network structure.

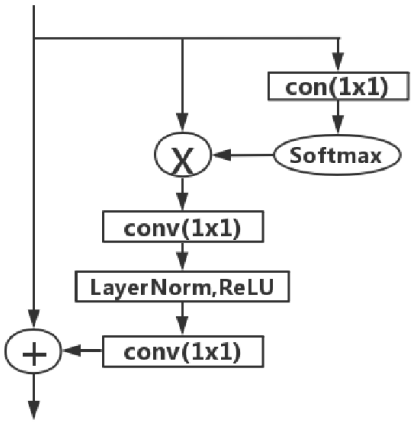

Fig. 2. Global context (GC) block.

In order to obtain multi-scale information, giving consideration to both semantic information and position information of small targets, we fuse the feature information of three different scales. The three outputs of YOLO v5 (y1, y2 and y3) derive from down sampling of different depths. They possess different semantic information and position information. Considering that majority target in our dataset are in a small size, motivated by Libra R-CNN [11], we use balanced feature pyramid to improve our model. As Figure 3 shows, we first operate up sampling and down sampling on y1 and y3 respectively, afterwards cat them in the channel dimension. For a better feature extraction, we embed the attention module. We use GC block to further refine the network. The architecture of GC block shows below in Figure 2. It can capture inter channel dependencies, so that beneficial to feature fusion. We then output the new y1', y2' and y3' with the operation of up sampling, down sampling and 1*1 convolution, respectively.

In order to verify our method, we conducted experiments on datasets. However, a new problem occurred. The convergence rate of the new model became very slow, and the optimal value was hard to obtained. Based on the idea of Resnet [7] , we further optimized our model. We enate the originate outputs y1, y2 and y3 with the new outputs y1', y2' and y3'. By these means, a) we protect the originate feature information, and fuse it with the new feature information. b) our model can promote optimization, speed up convergence and prevent the situation of no convergence.

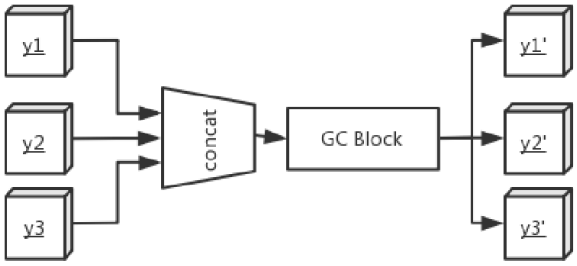

Fig. 3. Improved module.

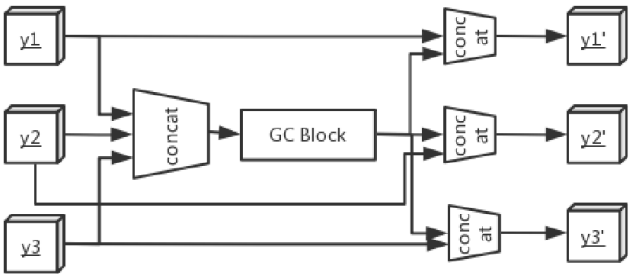

Fig. 4. Further improved module.

4 Experiments& results

4.1 Datasets & equipment

In order to verify our proposed method, we have conducted a lot of comparative experiments on the challenging dataset Tsinghua-Tencent-100K [7] (TT100K). The TT100k dataset contains almost 10, 000 pictures, and all of them are in the size of 2048*2048 pixels, while majority traffic sign are in a small size. In order to achieve better training and prediction results, we selected those 45 classes with more than 100 samples.

The experiments were run on a computer with Intel(R) Xeon (R) CPU, 32GB main memory and one Nvidia Quadro RTX5000 GPU with 16GB memory. The implementation environment is under the Pytorch1. 8. 1.

4.2 Results

For evaluating the effect of proposed model, we use recall, precision and mAP to quantitatively analyze our model. First, we compare improved YOLO v5 with the original YOLO v5. We list the comparative data in table 1.

Table 1. Comparison with the original method.

	P	**R**	**mAP@. 5**	**mAP@. 5:0. 95**
YOLOv5	0.85	0.828	87.8%	67.2%
Ours	0.874	0.861	89.7%	69.3%

From table 1, we can conclude that compared with original YOLO v5 network, the precision of our model increases by 2.4%, and the recall increases by 3.3%. Also, mAP@.5and mAP@. 5:0.95 increase by 1.9%, 2.1%, respectively.

In addition, we compare our methods with the state-of-the-art object detection methods. The results are shown in table 2.

Table 2. Comparison with other methods.

	SSD300 [10]	Faster RCNN [9]	YOLOv3	YOLOv4	YOLOv5	Ours
mAP@.5	63.71%	79.1%	82.4%	86.8%	87.8%	89.7%

In table 2, our method gets the best results on the TT100K dataset. Therefore, we can conclude that our method is effective on the dataset, and has made progress compared with the original method.

5 Conclusion

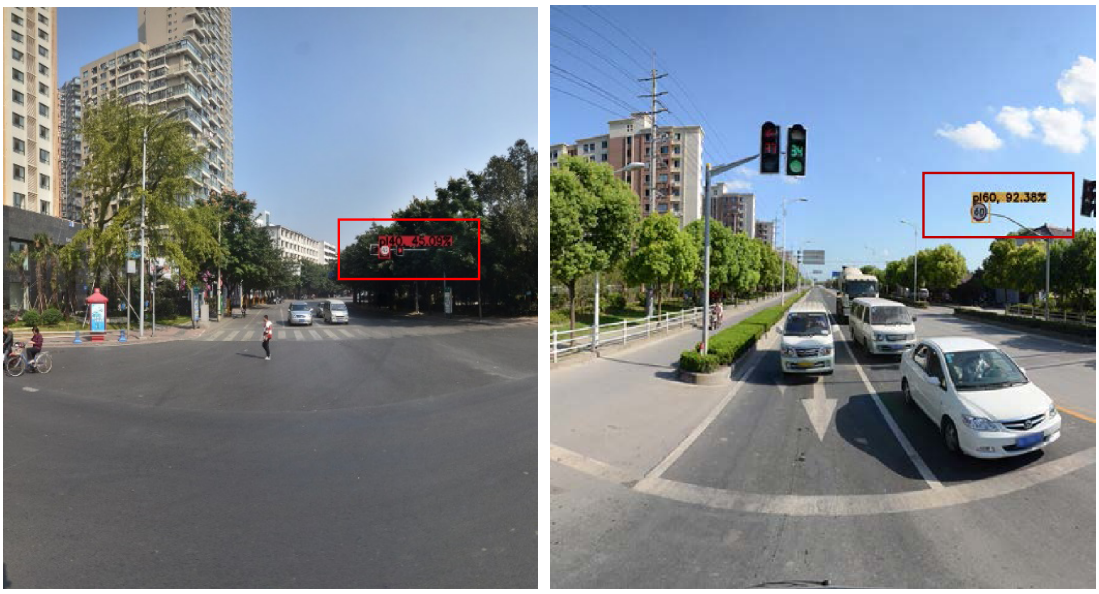

Fig. 5. Detection results in TT100K dataset. The detection results of small targets are marked with red boxes.

In this paper, we proposed an improved YOLO v5 to solve the problems existing in traffic sign detection. Aiming at traffic sign in small size, we chose the TT100K dataset. By comparing it with state-of-the-art methods mentioned above, our methods are better in accuracy. In the future, we will keep attempting to modify our model, so as to more suitable for automatic driving.

References

1. Redmon J, Divvala S, Girshick R, et al. You Only Look Once: Unified, Real-Time Object Detection[J]. Computer Vision & Pattern Recognition, 2016.
2. Liu W, Anguelov D, Erhan D , et al. SSD: Single Shot MultiBox Detector[J]. European Conference on Computer Vision, 2016.
3. Girshick R, Donahue J, Darrell T, et al. Rich feature hierarchies for accurate object detection and semantic segmentation[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2014: 580-587.

4. Felzenszwalb P F, Mcallester D A, Ramanan D . A discriminatively trained, multiscale, deformable part model[C]// 2008 IEEE Conference on Computer Vision and Pattern Recognition. IEEE, 2008.
5. Girshick R. Fast r-cnn[C]//Proceedings of the IEEE international conference on computer vision. 2015: 1440-1448.
6. Ren S, He K, Girshick R, et al. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks[J]. IEEE Transactions on Pattern Analysis & Machine Intelligence, 2017, 39(6):1137-1149.
7. He K, Zhang X, Ren S, et al. Deep residual learning for image recognition[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 770-778.
8. Zhe Z, Liang D, Zhang S, et al. Traffic-Sign Detection and Classification in the Wild[C]// 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, 2016.
9. Yao Z, Song X, Zhao L, et al. Realtime method for traffic sign detection and recognition based on YOLOv3tiny with multiscale feature extraction[J]. Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering, 2021, 235(7): 1978-1991.
10. Pan W, Liu B, Chen Y, et al. Traffic sign detection and recognition based on YOLO v3[J]. Transducer and Microsystem Technologies, 2019.
11. Pang J, Chen K, Shi J, et al. Libra R-CNN: Towards Balanced Learning for Object Detection[C]// 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, 2020.

Appendix B: Sample Code

Image detection-yoloV5

```
Import argparse

import os

import platform

import sys

from pathlib import Path


import torch


FILE = Path(__file__).resolve()

ROOT = FILE.parents[0]  # YOLOv5 root directory

if str(ROOT) not in sys.path:

    sys.path.append(str(ROOT))  # add ROOT to PATH

ROOT = Path(os.path.relpath(ROOT, Path.cwd()))  # relative


from models.common import DetectMultiBackend

from utils.dataloaders import IMG_FORMATS, VID_FORMATS, LoadImages, LoadScreenshots,
LoadStreams

from utils.general import (LOGGER, Profile, check_file, check_img_size, check_imshow,
check_requirements, colorstr, cv2,

                           increment_path, non_max_suppression, print_args, scale_boxes, strip_optimizer,
xyxy2xywh)

from utils.plots import Annotator, colors, save_one_box

from utils.torch_utils import select_device, smart_inference_mode



@smart_inference_mode()

def run(
```

```
    weights=ROOT / 'yolov5s.pt',  # model path or triton URL
    source=ROOT / 'data/images',  # file/dir/URL/glob/screen/0(webcam)
    data=ROOT / 'data/coco128.yaml',  # dataset.yaml path
    imgsz=(640, 640),  # inference size (height, width)
    conf_thres=0.25,  # confidence threshold
    iou_thres=0.45,  # NMS IOU threshold
    max_det=1000,  # maximum detections per image
    device='',  # cuda device, i.e. 0 or 0,1,2,3 or cpu
    view_img=False,  # show results
    save_txt=False,  # save results to *.txt
    save_conf=False,  # save confidences in --save-txt labels
    save_crop=False,  # save cropped prediction boxes
    nosave=False,  # do not save images/videos
    classes=None,  # filter by class: --class 0, or --class 0 2 3
    agnostic_nms=False,  # class-agnostic NMS
    augment=False,  # augmented inference
    visualize=False,  # visualize features
    update=False,  # update all models
    project=ROOT / 'runs/detect',  # save results to project/name
    name='exp',  # save results to project/name
    exist_ok=False,  # existing project/name ok, do not increment
    line_thickness=3,  # bounding box thickness (pixels)
    hide_labels=False,  # hide labels
    hide_conf=False,  # hide confidences
    half=False,  # use FP16 half-precision inference
    dnn=False,  # use OpenCV DNN for ONNX inference
    vid_stride=1,  # video frame-rate stride
):
    source = str(source)
    save_img = not nosave and not source.endswith('.txt')  # save inference images
    is_file = Path(source).suffix[1:] in (IMG_FORMATS + VID_FORMATS)
```

```
is_url = source.lower().startswith(('rtsp://', 'rtmp://', 'http://', 'https://'))
webcam = source.isnumeric() or source.endswith('.streams') or (is_url and not is_file)
screenshot = source.lower().startswith('screen')
if is_url and is_file:
    source = check_file(source)  # download

# Directories
save_dir = increment_path(Path(project) / name, exist_ok=exist_ok)  # increment run
(save_dir / 'labels' if save_txt else save_dir).mkdir(parents=True, exist_ok=True)  # make dir

# Load model
device = select_device(device)
model = DetectMultiBackend(weights, device=device, dnn=dnn, data=data, fp16=half)
stride, names, pt = model.stride, model.names, model.pt
imgsz = check_img_size(imgsz, s=stride)  # check image size

# Dataloader
bs = 1  # batch_size
if webcam:
    view_img = check_imshow(warn=True)
    dataset = LoadStreams(source, img_size=imgsz, stride=stride, auto=pt, vid_stride=vid_stride)
    bs = len(dataset)
elif screenshot:
    dataset = LoadScreenshots(source, img_size=imgsz, stride=stride, auto=pt)
else:
    dataset = LoadImages(source, img_size=imgsz, stride=stride, auto=pt, vid_stride=vid_stride)
vid_path, vid_writer = [None] * bs, [None] * bs

# Run inference
model.warmup(imgsz=(1 if pt or model.triton else bs, 3, *imgsz))  # warmup
seen, windows, dt = 0, [], (Profile(), Profile(), Profile())
```

```
    for path, im, im0s, vid_cap, s in dataset:
        with dt[0]:
            im = torch.from_numpy(im).to(model.device)
            im = im.half() if model.fp16 else im.float()  # uint8 to fp16/32
            im /= 255  # 0 - 255 to 0.0 - 1.0
            if len(im.shape) == 3:
                im = im[None]  # expand for batch dim

        # Inference
        with dt[1]:
            visualize = increment_path(save_dir / Path(path).stem, mkdir=True) if visualize else False
            pred = model(im, augment=augment, visualize=visualize)

        # NMS
        with dt[2]:
            pred = non_max_suppression(pred, conf_thres, iou_thres, classes, agnostic_nms,
max_det=max_det)

        # Second-stage classifier (optional)
        # pred = utils.general.apply_classifier(pred, classifier_model, im, im0s)

        # Process predictions
        for i, det in enumerate(pred):  # per image
            seen += 1
            if webcam:  # batch_size >= 1
                p, im0, frame = path[i], im0s[i].copy(), dataset.count
                s += f'{i}: '
            else:
                p, im0, frame = path, im0s.copy(), getattr(dataset, 'frame', 0)

            p = Path(p)  # to Path
```

```
        save_path = str(save_dir / p.name)  # im.jpg
        txt_path = str(save_dir / 'labels' / p.stem) + ('' if dataset.mode == 'image' else f'_{frame}')  # im.txt
        s += '%gx%g ' % im.shape[2:]  # print string
        gn = torch.tensor(im0.shape)[[1, 0, 1, 0]]  # normalization gain whwh
        imc = im0.copy() if save_crop else im0  # for save_crop
        annotator = Annotator(im0, line_width=line_thickness, example=str(names))
        if len(det):
            # Rescale boxes from img_size to im0 size
            det[:, :4] = scale_boxes(im.shape[2:], det[:, :4], im0.shape).round()

            # Print results
            for c in det[:, 5].unique():
                n = (det[:, 5] == c).sum()  # detections per class
                s += f"{n} {names[int(c)]}{'s' * (n > 1)}, "  # add to string

            # Write results
            for *xyxy, conf, cls in reversed(det):
                if save_txt:  # Write to file
                    xywh = (xyxy2xywh(torch.tensor(xyxy).view(1, 4)) / gn).view(-1).tolist()  # normalized xywh
                    line = (cls, *xywh, conf) if save_conf else (cls, *xywh)  # label format
                    with open(f'{txt_path}.txt', 'a') as f:
                        f.write(('%g ' * len(line)).rstrip() % line + '\n')

                if save_img or save_crop or view_img:  # Add bbox to image
                    c = int(cls)  # integer class
                    label = None if hide_labels else (names[c] if hide_conf else f'{names[c]} {conf:.2f}')
                    annotator.box_label(xyxy, label, color=colors(c, True))
                if save_crop:
                    save_one_box(xyxy, imc, file=save_dir / 'crops' / names[c] / f'{p.stem}.jpg', BGR=True)
```

```
        # Stream results
        im0 = annotator.result()
        if view_img:
            if platform.system() == 'Linux' and p not in windows:
                windows.append(p)
                cv2.namedWindow(str(p), cv2.WINDOW_NORMAL | cv2.WINDOW_KEEPRATIO)  # allow window resize (Linux)
                cv2.resizeWindow(str(p), im0.shape[1], im0.shape[0])
            cv2.imshow(str(p), im0)
            cv2.waitKey(1)  # 1 millisecond

        # Save results (image with detections)
        if save_img:
            if dataset.mode == 'image':
                cv2.imwrite(save_path, im0)
            else:  # 'video' or 'stream'
                if vid_path[i] != save_path:  # new video
                    vid_path[i] = save_path
                    if isinstance(vid_writer[i], cv2.VideoWriter):
                        vid_writer[i].release()  # release previous video writer
                    if vid_cap:  # video
                        fps = vid_cap.get(cv2.CAP_PROP_FPS)
                        w = int(vid_cap.get(cv2.CAP_PROP_FRAME_WIDTH))
                        h = int(vid_cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
                    else:  # stream
                        fps, w, h = 30, im0.shape[1], im0.shape[0]
                    save_path = str(Path(save_path).with_suffix('.mp4'))  # force *.mp4 suffix on results videos
                    vid_writer[i] = cv2.VideoWriter(save_path, cv2.VideoWriter_fourcc(*'mp4v'), fps, (w, h))
                vid_writer[i].write(im0)
```

```
        # Print time (inference-only)
        LOGGER.info(f"{s}{'' if len(det) else '(no detections), '}{dt[1].dt * 1E3:.1f}ms")

    # Print results
    t = tuple(x.t / seen * 1E3 for x in dt)  # speeds per image
    LOGGER.info(f'Speed: %.1fms pre-process, %.1fms inference, %.1fms NMS per image at shape {(1, 3, *imgsz)}' % t)
    if save_txt or save_img:
        s = f"\n{len(list(save_dir.glob('labels/*.txt')))} labels saved to {save_dir / 'labels'}" if save_txt else ''
        LOGGER.info(f"Results saved to {colorstr('bold', save_dir)}{s}")
    if update:
        strip_optimizer(weights[0])  # update model (to fix SourceChangeWarning)


def parse_opt():
    parser = argparse.ArgumentParser()
    parser.add_argument('--weights', nargs='+', type=str, default=ROOT / 'yolov5s.pt', help='model path or triton URL')
    parser.add_argument('--source', type=str, default=ROOT / 'data/images', help='file/dir/URL/glob/screen/0(webcam)')
    parser.add_argument('--data', type=str, default=ROOT / 'data/coco128.yaml', help='(optional) dataset.yaml path')
    parser.add_argument('--imgsz', '--img', '--img-size', nargs='+', type=int, default=[640], help='inference size h,w')
    parser.add_argument('--conf-thres', type=float, default=0.25, help='confidence threshold')
    parser.add_argument('--iou-thres', type=float, default=0.45, help='NMS IoU threshold')
    parser.add_argument('--max-det', type=int, default=1000, help='maximum detections per image')
    parser.add_argument('--device', default='', help='cuda device, i.e. 0 or 0,1,2,3 or cpu')
    parser.add_argument('--view-img', action='store_true', help='show results')
    parser.add_argument('--save-txt', action='store_true', help='save results to *.txt')
    parser.add_argument('--save-conf', action='store_true', help='save confidences in --save-txt labels')
```

```
    parser.add_argument('--save-crop', action='store_true', help='save cropped prediction boxes')
    parser.add_argument('--nosave', action='store_true', help='do not save images/videos')
    parser.add_argument('--classes', nargs='+', type=int, help='filter by class: --classes 0, or --classes 0 2 3')
    parser.add_argument('--agnostic-nms', action='store_true', help='class-agnostic NMS')
    parser.add_argument('--augment', action='store_true', help='augmented inference')
    parser.add_argument('--visualize', action='store_true', help='visualize features')
    parser.add_argument('--update', action='store_true', help='update all models')
    parser.add_argument('--project', default=ROOT / 'runs/detect', help='save results to project/name')
    parser.add_argument('--name', default='exp', help='save results to project/name')
    parser.add_argument('--exist-ok', action='store_true', help='existing project/name ok, do not increment')
    parser.add_argument('--line-thickness', default=3, type=int, help='bounding box thickness (pixels)')
    parser.add_argument('--hide-labels', default=False, action='store_true', help='hide labels')
    parser.add_argument('--hide-conf', default=False, action='store_true', help='hide confidences')
    parser.add_argument('--half', action='store_true', help='use FP16 half-precision inference')
    parser.add_argument('--dnn', action='store_true', help='use OpenCV DNN for ONNX inference')
    parser.add_argument('--vid-stride', type=int, default=1, help='video frame-rate stride')
    opt = parser.parse_args()
    opt.imgsz *= 2 if len(opt.imgsz) == 1 else 1  # expand
    print_args(vars(opt))
    return opt


def main(opt):
    check_requirements(ROOT / 'requirements.txt', exclude=('tensorboard', 'thop'))
    run(**vars(opt))


if __name__ == '__main__':
```

```
    opt = parse_opt()
    main(opt)
```

Training-yoloV5

```
import argparse
import math
import os
import random
import subprocess
import sys
import time
from copy import deepcopy
from datetime import datetime
from pathlib import Path

try:
    import comet_ml  # must be imported before torch (if installed)
except ImportError:
    comet_ml = None

import numpy as np
import torch
import torch.distributed as dist
import torch.nn as nn
import yaml
from torch.optim import lr_scheduler
```

```
from tqdm import tqdm


FILE = Path(__file__).resolve()
ROOT = FILE.parents[0]  # YOLOv5 root directory
if str(ROOT) not in sys.path:
    sys.path.append(str(ROOT))  # add ROOT to PATH
ROOT = Path(os.path.relpath(ROOT, Path.cwd()))  # relative


import val as validate  # for end-of-epoch mAP
from models.experimental import attempt_load
from models.yolo import Model
from utils.autoanchor import check_anchors
from utils.autobatch import check_train_batch_size
from utils.callbacks import Callbacks
from utils.dataloaders import create_dataloader
from utils.downloads import attempt_download, is_url
from utils.general import (LOGGER, TQDM_BAR_FORMAT, check_amp, check_dataset, check_file,
check_git_info,
                           check_git_status, check_img_size, check_requirements, check_suffix, check_yaml,
colorstr,
                           get_latest_run, increment_path, init_seeds, intersect_dicts, labels_to_class_weights,
                           labels_to_image_weights, methods, one_cycle, print_args, print_mutation,
strip_optimizer,
                           yaml_save)
from utils.loggers import Loggers
from utils.loggers.comet.comet_utils import check_comet_resume
from utils.loss import ComputeLoss
from utils.metrics import fitness
from utils.plots import plot_evolve
from utils.torch_utils import (EarlyStopping, ModelEMA, de_parallel, select_device, smart_DDP,
smart_optimizer,
                               smart_resume, torch_distributed_zero_first)
```

```
LOCAL_RANK = int(os.getenv('LOCAL_RANK', -1))  # https://pytorch.org/docs/stable/elastic/run.html
RANK = int(os.getenv('RANK', -1))
WORLD_SIZE = int(os.getenv('WORLD_SIZE', 1))
GIT_INFO = check_git_info()



def train(hyp, opt, device, callbacks):  # hyp is path/to/hyp.yaml or hyp dictionary
    save_dir, epochs, batch_size, weights, single_cls, evolve, data, cfg, resume, noval, nosave, workers, freeze = \
        Path(opt.save_dir), opt.epochs, opt.batch_size, opt.weights, opt.single_cls, opt.evolve, opt.data, opt.cfg, \
        opt.resume, opt.noval, opt.nosave, opt.workers, opt.freeze
    callbacks.run('on_pretrain_routine_start')

    # Directories
    w = save_dir / 'weights'  # weights dir
    (w.parent if evolve else w).mkdir(parents=True, exist_ok=True)  # make dir
    last, best = w / 'last.pt', w / 'best.pt'

    # Hyperparameters
    if isinstance(hyp, str):
        with open(hyp, errors='ignore') as f:
            hyp = yaml.safe_load(f)  # load hyps dict
    LOGGER.info(colorstr('hyperparameters: ') + ', '.join(f'{k}={v}' for k, v in hyp.items()))
    opt.hyp = hyp.copy()  # for saving hyps to checkpoints

    # Save run settings
    if not evolve:
        yaml_save(save_dir / 'hyp.yaml', hyp)
        yaml_save(save_dir / 'opt.yaml', vars(opt))
```

```
    # Loggers
    data_dict = None
    if RANK in {-1, 0}:
        loggers = Loggers(save_dir, weights, opt, hyp, LOGGER)  # loggers instance

        # Register actions
        for k in methods(loggers):
            callbacks.register_action(k, callback=getattr(loggers, k))

        # Process custom dataset artifact link
        data_dict = loggers.remote_dataset
        if resume:  # If resuming runs from remote artifact
            weights, epochs, hyp, batch_size = opt.weights, opt.epochs, opt.hyp, opt.batch_size

    # Config
    plots = not evolve and not opt.noplots  # create plots
    cuda = device.type != 'cpu'
    init_seeds(opt.seed + 1 + RANK, deterministic=True)
    with torch_distributed_zero_first(LOCAL_RANK):
        data_dict = data_dict or check_dataset(data)  # check if None
    train_path, val_path = data_dict['train'], data_dict['val']
    nc = 1 if single_cls else int(data_dict['nc'])  # number of classes
    names = {0: 'item'} if single_cls and len(data_dict['names']) != 1 else data_dict['names']  # class
names
    is_coco = isinstance(val_path, str) and val_path.endswith('coco/val2017.txt')  # COCO dataset

    # Model
    check_suffix(weights, '.pt')  # check weights
    pretrained = weights.endswith('.pt')
    if pretrained:
        with torch_distributed_zero_first(LOCAL_RANK):
```

```
            weights = attempt_download(weights)  # download if not found locally
        ckpt = torch.load(weights, map_location='cpu')  # load checkpoint to CPU to avoid CUDA memory leak
        model = Model(cfg or ckpt['model'].yaml, ch=3, nc=nc, anchors=hyp.get('anchors')).to(device)  # create
        exclude = ['anchor'] if (cfg or hyp.get('anchors')) and not resume else []  # exclude keys
        csd = ckpt['model'].float().state_dict()  # checkpoint state_dict as FP32
        csd = intersect_dicts(csd, model.state_dict(), exclude=exclude)  # intersect
        model.load_state_dict(csd, strict=False)  # load
        LOGGER.info(f'Transferred {len(csd)}/{len(model.state_dict())} items from {weights}')  # report
    else:
        model = Model(cfg, ch=3, nc=nc, anchors=hyp.get('anchors')).to(device)  # create
    amp = check_amp(model)  # check AMP

    # Freeze
    freeze = [f'model.{x}.' for x in (freeze if len(freeze) > 1 else range(freeze[0]))]  # layers to freeze
    for k, v in model.named_parameters():
        v.requires_grad = True  # train all layers
        # v.register_hook(lambda x: torch.nan_to_num(x))  # NaN to 0 (commented for erratic training results)
        if any(x in k for x in freeze):
            LOGGER.info(f'freezing {k}')
            v.requires_grad = False

    # Image size
    gs = max(int(model.stride.max()), 32)  # grid size (max stride)
    imgsz = check_img_size(opt.imgsz, gs, floor=gs * 2)  # verify imgsz is gs-multiple

    # Batch size
    if RANK == -1 and batch_size == -1:  # single-GPU only, estimate best batch size
        batch_size = check_train_batch_size(model, imgsz, amp)
        loggers.on_params_update({'batch_size': batch_size})
```

```
    # Optimizer
    nbs = 64  # nominal batch size
    accumulate = max(round(nbs / batch_size), 1)  # accumulate loss before optimizing
    hyp['weight_decay'] *= batch_size * accumulate / nbs  # scale weight_decay
    optimizer = smart_optimizer(model, opt.optimizer, hyp['lr0'], hyp['momentum'],
hyp['weight_decay'])

    # Scheduler
    if opt.cos_lr:
        lf = one_cycle(1, hyp['lrf'], epochs)  # cosine 1->hyp['lrf']
    else:
        lf = lambda x: (1 - x / epochs) * (1.0 - hyp['lrf']) + hyp['lrf']  # linear
    scheduler = lr_scheduler.LambdaLR(optimizer, lr_lambda=lf)  # plot_lr_scheduler(optimizer,
scheduler, epochs)

    # EMA
    ema = ModelEMA(model) if RANK in {-1, 0} else None

    # Resume
    best_fitness, start_epoch = 0.0, 0
    if pretrained:
        if resume:
            best_fitness, start_epoch, epochs = smart_resume(ckpt, optimizer, ema, weights, epochs,
resume)
        del ckpt, csd

    # DP mode
    if cuda and RANK == -1 and torch.cuda.device_count() > 1:
        LOGGER.warning(
            'WARNING ⚠ DP not recommended, use torch.distributed.run for best DDP Multi-GPU
results.\n'
```

```
        'See Multi-GPU Tutorial at https://docs.ultralytics.com/yolov5/tutorials/multi_gpu_training to get started.'
    )
    model = torch.nn.DataParallel(model)

# SyncBatchNorm
if opt.sync_bn and cuda and RANK != -1:
    model = torch.nn.SyncBatchNorm.convert_sync_batchnorm(model).to(device)
    LOGGER.info('Using SyncBatchNorm()')

# Trainloader
train_loader, dataset = create_dataloader(train_path,
                                          imgsz,
                                          batch_size // WORLD_SIZE,
                                          gs,
                                          single_cls,
                                          hyp=hyp,
                                          augment=True,
                                          cache=None if opt.cache == 'val' else opt.cache,
                                          rect=opt.rect,
                                          rank=LOCAL_RANK,
                                          workers=workers,
                                          image_weights=opt.image_weights,
                                          quad=opt.quad,
                                          prefix=colorstr('train: '),
                                          shuffle=True,
                                          seed=opt.seed)
labels = np.concatenate(dataset.labels, 0)
mlc = int(labels[:, 0].max())  # max label class
assert mlc < nc, f'Label class {mlc} exceeds nc={nc} in {data}. Possible class labels are 0-{nc - 1}'
```

```
    # Process 0
    if RANK in {-1, 0}:
        val_loader = create_dataloader(val_path,
                                       imgsz,
                                       batch_size // WORLD_SIZE * 2,
                                       gs,
                                       single_cls,
                                       hyp=hyp,
                                       cache=None if noval else opt.cache,
                                       rect=True,
                                       rank=-1,
                                       workers=workers * 2,
                                       pad=0.5,
                                       prefix=colorstr('val: '))[0]

        if not resume:
            if not opt.noautoanchor:
                check_anchors(dataset, model=model, thr=hyp['anchor_t'], imgsz=imgsz)  # run
AutoAnchor
            model.half().float()  # pre-reduce anchor precision

        callbacks.run('on_pretrain_routine_end', labels, names)

    # DDP mode
    if cuda and RANK != -1:
        model = smart_DDP(model)

    # Model attributes
    nl = de_parallel(model).model[-1].nl  # number of detection layers (to scale hyps)
    hyp['box'] *= 3 / nl  # scale to layers
    hyp['cls'] *= nc / 80 * 3 / nl  # scale to classes and layers
```

```
    hyp['obj'] *= (imgsz / 640) ** 2 * 3 / nl  # scale to image size and layers
    hyp['label_smoothing'] = opt.label_smoothing
    model.nc = nc  # attach number of classes to model
    model.hyp = hyp  # attach hyperparameters to model
    model.class_weights = labels_to_class_weights(dataset.labels, nc).to(device) * nc  # attach class weights
    model.names = names

    # Start training
    t0 = time.time()
    nb = len(train_loader)  # number of batches
    nw = max(round(hyp['warmup_epochs'] * nb), 100)  # number of warmup iterations, max(3 epochs, 100 iterations)
    # nw = min(nw, (epochs - start_epoch) / 2 * nb)  # limit warmup to < 1/2 of training
    last_opt_step = -1
    maps = np.zeros(nc)  # mAP per class
    results = (0, 0, 0, 0, 0, 0, 0)  # P, R, mAP@.5, mAP@.5-.95, val_loss(box, obj, cls)
    scheduler.last_epoch = start_epoch - 1  # do not move
    scaler = torch.cuda.amp.GradScaler(enabled=amp)
    stopper, stop = EarlyStopping(patience=opt.patience), False
    compute_loss = ComputeLoss(model)  # init loss class
    callbacks.run('on_train_start')
    LOGGER.info(f'Image sizes {imgsz} train, {imgsz} val\n'
                f'Using {train_loader.num_workers * WORLD_SIZE} dataloader workers\n'
                f"Logging results to {colorstr('bold', save_dir)}\n"
                f'Starting training for {epochs} epochs...')
    for epoch in range(start_epoch, epochs):  # epoch ------------------------------------------------------------------
        callbacks.run('on_train_epoch_start')
        model.train()

        # Update image weights (optional, single-GPU only)
```

```
    if opt.image_weights:
        cw = model.class_weights.cpu().numpy() * (1 - maps) ** 2 / nc  # class weights
        iw = labels_to_image_weights(dataset.labels, nc=nc, class_weights=cw)  # image weights
        dataset.indices = random.choices(range(dataset.n), weights=iw, k=dataset.n)  # rand weighted idx

    # Update mosaic border (optional)
    # b = int(random.uniform(0.25 * imgsz, 0.75 * imgsz + gs) // gs * gs)
    # dataset.mosaic_border = [b - imgsz, -b]  # height, width borders

    mloss = torch.zeros(3, device=device)  # mean losses
    if RANK != -1:
        train_loader.sampler.set_epoch(epoch)
    pbar = enumerate(train_loader)
    LOGGER.info(('\n' + '%11s' * 7) % ('Epoch', 'GPU_mem', 'box_loss', 'obj_loss', 'cls_loss', 'Instances', 'Size'))
    if RANK in {-1, 0}:
        pbar = tqdm(pbar, total=nb, bar_format=TQDM_BAR_FORMAT)  # progress bar
    optimizer.zero_grad()
    for i, (imgs, targets, paths, _) in pbar:  # batch -------------------------------------------------------------
        callbacks.run('on_train_batch_start')
        ni = i + nb * epoch  # number integrated batches (since train start)
        imgs = imgs.to(device, non_blocking=True).float() / 255  # uint8 to float32, 0-255 to 0.0-1.0

        # Warmup
        if ni <= nw:
            xi = [0, nw]  # x interp
            # compute_loss.gr = np.interp(ni, xi, [0.0, 1.0])  # iou loss ratio (obj_loss = 1.0 or iou)
            accumulate = max(1, np.interp(ni, xi, [1, nbs / batch_size]).round())
            for j, x in enumerate(optimizer.param_groups):
                # bias lr falls from 0.1 to lr0, all other lrs rise from 0.0 to lr0
                x['lr'] = np.interp(ni, xi, [hyp['warmup_bias_lr'] if j == 0 else 0.0, x['initial_lr'] * lf(epoch)])
```

```
            if 'momentum' in x:
                x['momentum'] = np.interp(ni, xi, [hyp['warmup_momentum'], hyp['momentum']])

        # Multi-scale
        if opt.multi_scale:
            sz = random.randrange(int(imgsz * 0.5), int(imgsz * 1.5) + gs) // gs * gs  # size
            sf = sz / max(imgs.shape[2:])  # scale factor
            if sf != 1:
                ns = [math.ceil(x * sf / gs) * gs for x in imgs.shape[2:]]  # new shape (stretched to gs-multiple)
                imgs = nn.functional.interpolate(imgs, size=ns, mode='bilinear', align_corners=False)

        # Forward
        with torch.cuda.amp.autocast(amp):
            pred = model(imgs)  # forward
            loss, loss_items = compute_loss(pred, targets.to(device))  # loss scaled by batch_size
            if RANK != -1:
                loss *= WORLD_SIZE  # gradient averaged between devices in DDP mode
            if opt.quad:
                loss *= 4.

        # Backward
        scaler.scale(loss).backward()

        # Optimize - https://pytorch.org/docs/master/notes/amp_examples.html
        if ni - last_opt_step >= accumulate:
            scaler.unscale_(optimizer)  # unscale gradients
            torch.nn.utils.clip_grad_norm_(model.parameters(), max_norm=10.0)  # clip gradients
            scaler.step(optimizer)  # optimizer.step
            scaler.update()
            optimizer.zero_grad()
```

```
            if ema:
                ema.update(model)
            last_opt_step = ni

        # Log
        if RANK in {-1, 0}:
            mloss = (mloss * i + loss_items) / (i + 1)  # update mean losses
            mem = f'{torch.cuda.memory_reserved() / 1E9 if torch.cuda.is_available() else 0:.3g}G'  #
(GB)
            pbar.set_description(('%11s' * 2 + '%11.4g' * 5) %
                                 (f'{epoch}/{epochs - 1}', mem, *mloss, targets.shape[0], imgs.shape[-1]))
            callbacks.run('on_train_batch_end', model, ni, imgs, targets, paths, list(mloss))
            if callbacks.stop_training:
                return
        # end batch ------------------------------------------------------------------------------------------------

    # Scheduler
    lr = [x['lr'] for x in optimizer.param_groups]  # for loggers
    scheduler.step()

    if RANK in {-1, 0}:
        # mAP
        callbacks.run('on_train_epoch_end', epoch=epoch)
        ema.update_attr(model, include=['yaml', 'nc', 'hyp', 'names', 'stride', 'class_weights'])
        final_epoch = (epoch + 1 == epochs) or stopper.possible_stop
        if not noval or final_epoch:  # Calculate mAP
            results, maps, _ = validate.run(data_dict,
                                            batch_size=batch_size // WORLD_SIZE * 2,
                                            imgsz=imgsz,
                                            half=amp,
                                            model=ema.ema,
```

```
                                single_cls=single_cls,
                                dataloader=val_loader,
                                save_dir=save_dir,
                                plots=False,
                                callbacks=callbacks,
                                compute_loss=compute_loss)

        # Update best mAP
        fi = fitness(np.array(results).reshape(1, -1))  # weighted combination of [P, R, mAP@.5, mAP@.5-.95]
        stop = stopper(epoch=epoch, fitness=fi)  # early stop check
        if fi > best_fitness:
            best_fitness = fi
        log_vals = list(mloss) + list(results) + lr
        callbacks.run('on_fit_epoch_end', log_vals, epoch, best_fitness, fi)

        # Save model
        if (not nosave) or (final_epoch and not evolve):  # if save
            ckpt = {
                'epoch': epoch,
                'best_fitness': best_fitness,
                'model': deepcopy(de_parallel(model)).half(),
                'ema': deepcopy(ema.ema).half(),
                'updates': ema.updates,
                'optimizer': optimizer.state_dict(),
                'opt': vars(opt),
                'git': GIT_INFO,  # {remote, branch, commit} if a git repo
                'date': datetime.now().isoformat()}

            # Save last, best and delete
            torch.save(ckpt, last)
```

```
            if best_fitness == fi:
                torch.save(ckpt, best)
            if opt.save_period > 0 and epoch % opt.save_period == 0:
                torch.save(ckpt, w / f'epoch{epoch}.pt')
            del ckpt
            callbacks.run('on_model_save', last, epoch, final_epoch, best_fitness, fi)

        # EarlyStopping
        if RANK != -1:  # if DDP training
            broadcast_list = [stop if RANK == 0 else None]
            dist.broadcast_object_list(broadcast_list, 0)  # broadcast 'stop' to all ranks
            if RANK != 0:
                stop = broadcast_list[0]
        if stop:
            break  # must break all DDP ranks

        # end epoch ----------------------------------------------------------------------------------------------------
    # end training -------------------------------------------------------------------------------------------------
    if RANK in {-1, 0}:
        LOGGER.info(f'\n{epoch - start_epoch + 1} epochs completed in {(time.time() - t0) / 3600:.3f} hours.')
        for f in last, best:
            if f.exists():
                strip_optimizer(f)  # strip optimizers
                if f is best:
                    LOGGER.info(f'\nValidating {f}...')
                    results, _, _ = validate.run(
                        data_dict,
                        batch_size=batch_size // WORLD_SIZE * 2,
                        imgsz=imgsz,
                        model=attempt_load(f, device).half(),
```

```
                    iou_thres=0.65 if is_coco else 0.60,  # best pycocotools at iou 0.65
                    single_cls=single_cls,
                    dataloader=val_loader,
                    save_dir=save_dir,
                    save_json=is_coco,
                    verbose=True,
                    plots=plots,
                    callbacks=callbacks,
                    compute_loss=compute_loss)  # val best model with plots
                if is_coco:
                    callbacks.run('on_fit_epoch_end', list(mloss) + list(results) + lr, epoch, best_fitness, fi)

        callbacks.run('on_train_end', last, best, epoch, results)

    torch.cuda.empty_cache()
    return results


def parse_opt(known=False):
    parser = argparse.ArgumentParser()
    parser.add_argument('--weights', type=str, default=ROOT / 'yolov5s.pt', help='initial weights path')
    parser.add_argument('--cfg', type=str, default='', help='model.yaml path')
    parser.add_argument('--data', type=str, default=ROOT / 'data/coco128.yaml', help='dataset.yaml path')
    parser.add_argument('--hyp', type=str, default=ROOT / 'data/hyps/hyp.scratch-low.yaml', help='hyperparameters path')
    parser.add_argument('--epochs', type=int, default=100, help='total training epochs')
    parser.add_argument('--batch-size', type=int, default=16, help='total batch size for all GPUs, -1 for autobatch')
    parser.add_argument('--imgsz', '--img', '--img-size', type=int, default=640, help='train, val image size (pixels)')
    parser.add_argument('--rect', action='store_true', help='rectangular training')
```

```
    parser.add_argument('--resume', nargs='?', const=True, default=False, help='resume most recent training')

    parser.add_argument('--nosave', action='store_true', help='only save final checkpoint')

    parser.add_argument('--noval', action='store_true', help='only validate final epoch')

    parser.add_argument('--noautoanchor', action='store_true', help='disable AutoAnchor')

    parser.add_argument('--noplots', action='store_true', help='save no plot files')

    parser.add_argument('--evolve', type=int, nargs='?', const=300, help='evolve hyperparameters for x generations')

    parser.add_argument('--bucket', type=str, default='', help='gsutil bucket')

    parser.add_argument('--cache', type=str, nargs='?', const='ram', help='image --cache ram/disk')

    parser.add_argument('--image-weights', action='store_true', help='use weighted image selection for training')

    parser.add_argument('--device', default='', help='cuda device, i.e. 0 or 0,1,2,3 or cpu')

    parser.add_argument('--multi-scale', action='store_true', help='vary img-size +/- 50%%')

    parser.add_argument('--single-cls', action='store_true', help='train multi-class data as single-class')

    parser.add_argument('--optimizer', type=str, choices=['SGD', 'Adam', 'AdamW'], default='SGD', help='optimizer')

    parser.add_argument('--sync-bn', action='store_true', help='use SyncBatchNorm, only available in DDP mode')

    parser.add_argument('--workers', type=int, default=8, help='max dataloader workers (per RANK in DDP mode)')

    parser.add_argument('--project', default=ROOT / 'runs/train', help='save to project/name')

    parser.add_argument('--name', default='exp', help='save to project/name')

    parser.add_argument('--exist-ok', action='store_true', help='existing project/name ok, do not increment')

    parser.add_argument('--quad', action='store_true', help='quad dataloader')

    parser.add_argument('--cos-lr', action='store_true', help='cosine LR scheduler')

    parser.add_argument('--label-smoothing', type=float, default=0.0, help='Label smoothing epsilon')

    parser.add_argument('--patience', type=int, default=100, help='EarlyStopping patience (epochs without improvement)')

    parser.add_argument('--freeze', nargs='+', type=int, default=[0], help='Freeze layers: backbone=10, first3=0 1 2')

    parser.add_argument('--save-period', type=int, default=-1, help='Save checkpoint every x epochs (disabled if < 1)')
```

```
    parser.add_argument('--seed', type=int, default=0, help='Global training seed')
    parser.add_argument('--local_rank', type=int, default=-1, help='Automatic DDP Multi-GPU argument, do not modify')

    # Logger arguments
    parser.add_argument('--entity', default=None, help='Entity')
    parser.add_argument('--upload_dataset', nargs='?', const=True, default=False, help='Upload data, "val" option')
    parser.add_argument('--bbox_interval', type=int, default=-1, help='Set bounding-box image logging interval')
    parser.add_argument('--artifact_alias', type=str, default='latest', help='Version of dataset artifact to use')

    return parser.parse_known_args()[0] if known else parser.parse_args()


def main(opt, callbacks=Callbacks()):
    # Checks
    if RANK in {-1, 0}:
        print_args(vars(opt))
        check_git_status()
        check_requirements(ROOT / 'requirements.txt')

    # Resume (from specified or most recent last.pt)
    if opt.resume and not check_comet_resume(opt) and not opt.evolve:
        last = Path(check_file(opt.resume) if isinstance(opt.resume, str) else get_latest_run())
        opt_yaml = last.parent.parent / 'opt.yaml'  # train options yaml
        opt_data = opt.data  # original dataset
        if opt_yaml.is_file():
            with open(opt_yaml, errors='ignore') as f:
                d = yaml.safe_load(f)
        else:
```

```
            d = torch.load(last, map_location='cpu')['opt']
        opt = argparse.Namespace(**d)  # replace
        opt.cfg, opt.weights, opt.resume = '', str(last), True  # reinstate
        if is_url(opt_data):
            opt.data = check_file(opt_data)  # avoid HUB resume auth timeout
    else:
        opt.data, opt.cfg, opt.hyp, opt.weights, opt.project = \
            check_file(opt.data), check_yaml(opt.cfg), check_yaml(opt.hyp), str(opt.weights),
str(opt.project)  # checks
        assert len(opt.cfg) or len(opt.weights), 'either --cfg or --weights must be specified'
        if opt.evolve:
            if opt.project == str(ROOT / 'runs/train'):  # if default project name, rename to runs/evolve
                opt.project = str(ROOT / 'runs/evolve')
            opt.exist_ok, opt.resume = opt.resume, False  # pass resume to exist_ok and disable resume
        if opt.name == 'cfg':
            opt.name = Path(opt.cfg).stem  # use model.yaml as name
        opt.save_dir = str(increment_path(Path(opt.project) / opt.name, exist_ok=opt.exist_ok))


    # DDP mode
    device = select_device(opt.device, batch_size=opt.batch_size)
    if LOCAL_RANK != -1:
        msg = 'is not compatible with YOLOv5 Multi-GPU DDP training'
        assert not opt.image_weights, f'--image-weights {msg}'
        assert not opt.evolve, f'--evolve {msg}'
        assert opt.batch_size != -1, f'AutoBatch with --batch-size -1 {msg}, please pass a valid --batch-
size'
        assert opt.batch_size % WORLD_SIZE == 0, f'--batch-size {opt.batch_size} must be multiple of
WORLD_SIZE'
        assert torch.cuda.device_count() > LOCAL_RANK, 'insufficient CUDA devices for DDP command'
        torch.cuda.set_device(LOCAL_RANK)
        device = torch.device('cuda', LOCAL_RANK)
        dist.init_process_group(backend='nccl' if dist.is_nccl_available() else 'gloo')
```

```
# Train
if not opt.evolve:
    train(opt.hyp, opt, device, callbacks)


# Evolve hyperparameters (optional)
else:
    # Hyperparameter evolution metadata (mutation scale 0-1, lower_limit, upper_limit)
    meta = {
        'lr0': (1, 1e-5, 1e-1),  # initial learning rate (SGD=1E-2, Adam=1E-3)
        'lrf': (1, 0.01, 1.0),  # final OneCycleLR learning rate (lr0 * lrf)
        'momentum': (0.3, 0.6, 0.98),  # SGD momentum/Adam beta1
        'weight_decay': (1, 0.0, 0.001),  # optimizer weight decay
        'warmup_epochs': (1, 0.0, 5.0),  # warmup epochs (fractions ok)
        'warmup_momentum': (1, 0.0, 0.95),  # warmup initial momentum
        'warmup_bias_lr': (1, 0.0, 0.2),  # warmup initial bias lr
        'box': (1, 0.02, 0.2),  # box loss gain
        'cls': (1, 0.2, 4.0),  # cls loss gain
        'cls_pw': (1, 0.5, 2.0),  # cls BCELoss positive_weight
        'obj': (1, 0.2, 4.0),  # obj loss gain (scale with pixels)
        'obj_pw': (1, 0.5, 2.0),  # obj BCELoss positive_weight
        'iou_t': (0, 0.1, 0.7),  # IoU training threshold
        'anchor_t': (1, 2.0, 8.0),  # anchor-multiple threshold
        'anchors': (2, 2.0, 10.0),  # anchors per output grid (0 to ignore)
        'fl_gamma': (0, 0.0, 2.0),  # focal loss gamma (efficientDet default gamma=1.5)
        'hsv_h': (1, 0.0, 0.1),  # image HSV-Hue augmentation (fraction)
        'hsv_s': (1, 0.0, 0.9),  # image HSV-Saturation augmentation (fraction)
        'hsv_v': (1, 0.0, 0.9),  # image HSV-Value augmentation (fraction)
        'degrees': (1, 0.0, 45.0),  # image rotation (+/- deg)
        'translate': (1, 0.0, 0.9),  # image translation (+/- fraction)
        'scale': (1, 0.0, 0.9),  # image scale (+/- gain)
```

```
    'shear': (1, 0.0, 10.0),  # image shear (+/- deg)
    'perspective': (0, 0.0, 0.001),  # image perspective (+/- fraction), range 0-0.001
    'flipud': (1, 0.0, 1.0),  # image flip up-down (probability)
    'fliplr': (0, 0.0, 1.0),  # image flip left-right (probability)
    'mosaic': (1, 0.0, 1.0),  # image mixup (probability)
    'mixup': (1, 0.0, 1.0),  # image mixup (probability)
    'copy_paste': (1, 0.0, 1.0)}  # segment copy-paste (probability)

with open(opt.hyp, errors='ignore') as f:
    hyp = yaml.safe_load(f)  # load hyps dict
    if 'anchors' not in hyp:  # anchors commented in hyp.yaml
        hyp['anchors'] = 3
if opt.noautoanchor:
    del hyp['anchors'], meta['anchors']
opt.noval, opt.nosave, save_dir = True, True, Path(opt.save_dir)  # only val/save final epoch
# ei = [isinstance(x, (int, float)) for x in hyp.values()]  # evolvable indices
evolve_yaml, evolve_csv = save_dir / 'hyp_evolve.yaml', save_dir / 'evolve.csv'
if opt.bucket:
    # download evolve.csv if exists
    subprocess.run([
        'gsutil',
        'cp',
        f'gs://{opt.bucket}/evolve.csv',
        str(evolve_csv), ])

for _ in range(opt.evolve):  # generations to evolve
    if evolve_csv.exists():  # if evolve.csv exists: select best hyps and mutate
        # Select parent(s)
        parent = 'single'  # parent selection method: 'single' or 'weighted'
        x = np.loadtxt(evolve_csv, ndmin=2, delimiter=',', skiprows=1)
        n = min(5, len(x))  # number of previous results to consider
```

```
    x = x[np.argsort(-fitness(x))][:n]  # top n mutations
    w = fitness(x) - fitness(x).min() + 1E-6  # weights (sum > 0)
    if parent == 'single' or len(x) == 1:
        # x = x[random.randint(0, n - 1)]  # random selection
        x = x[random.choices(range(n), weights=w)[0]]  # weighted selection
    elif parent == 'weighted':
        x = (x * w.reshape(n, 1)).sum(0) / w.sum()  # weighted combination

    # Mutate
    mp, s = 0.8, 0.2  # mutation probability, sigma
    npr = np.random
    npr.seed(int(time.time()))
    g = np.array([meta[k][0] for k in hyp.keys()])  # gains 0-1
    ng = len(meta)
    v = np.ones(ng)
    while all(v == 1):  # mutate until a change occurs (prevent duplicates)
        v = (g * (npr.random(ng) < mp) * npr.randn(ng) * npr.random() * s + 1).clip(0.3, 3.0)
    for i, k in enumerate(hyp.keys()):  # plt.hist(v.ravel(), 300)
        hyp[k] = float(x[i + 7] * v[i])  # mutate

# Constrain to limits
for k, v in meta.items():
    hyp[k] = max(hyp[k], v[1])  # lower limit
    hyp[k] = min(hyp[k], v[2])  # upper limit
    hyp[k] = round(hyp[k], 5)  # significant digits

# Train mutation
results = train(hyp.copy(), opt, device, callbacks)
callbacks = Callbacks()
# Write mutation results
```

```
            keys = ('metrics/precision', 'metrics/recall', 'metrics/mAP_0.5', 'metrics/mAP_0.5:0.95',
'val/box_loss',
                    'val/obj_loss', 'val/cls_loss')
            print_mutation(keys, results, hyp.copy(), save_dir, opt.bucket)

        # Plot results
        plot_evolve(evolve_csv)
        LOGGER.info(f'Hyperparameter evolution finished {opt.evolve} generations\n'
                    f"Results saved to {colorstr('bold', save_dir)}\n"
                    f'Usage example: $ python train.py --hyp {evolve_yaml}')


def run(**kwargs):
    # Usage: import train; train.run(data='coco128.yaml', imgsz=320, weights='yolov5m.pt')
    opt = parse_opt(True)
    for k, v in kwargs.items():
        setattr(opt, k, v)
    main(opt)
    return opt


if __name__ == '__main__':
    opt = parse_opt()
    main(opt)
```

Appendix C: CO-PO And CO-PSO Mapping

COURSE OUTCOMES:

After completion of the course the student will be able to

SL. NO	DESCRIPTION	Blooms' Taxonomy Level
CO1	Identify technically and economically feasible problems (Cognitive Knowledge Level: Apply)	Level 3: Apply
CO2	Identify and survey the relevant literature for getting exposed to related solutions and get familiarized with software development processes (Cognitive Knowledge Level: Apply)	Level 3: Apply
CO3	Perform requirement analysis, identify design methodologies and develop adaptable & reusable solutions of minimal complexity by using modern tools & advanced programming techniques (Cognitive Knowledge Level: Apply)	Level 3: Apply
CO4	Prepare technical report and deliver presentation (Cognitive Knowledge Level: Apply)	Level 3: Apply
CO5	Apply engineering and management principles to achieve the goal of the project (Cognitive Knowledge Level: Apply)	Level 3: Apply

CO-PO AND CO-PSO MAPPING

	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO 10	PO 11	PO 12	PSO 1	PSO 2	PSO3
CO1	3	3	3	3		2	2	3	2	2	2	3	2	2	2
CO2	3	3	3	3	3	2		3	2	3	2	3	2	2	2
CO3	3	3	3	3	3	2	2	3	2	2	2	3			2
CO4	2	3	2	2	2			3	3	3	2	3	2	2	2
CO5	3	3	3	2	2	2	2	3	2		2	3	2	2	2

3/2/1: high/medium/low

JUSTIFICATIONS FOR CO-PO MAPPING

MAPPING	LOW/ MEDIUM/ HIGH	JUSTIFICATION
100003/CS622T.1-PO1	**HIGH**	Identify technically and economically feasible problems by applying the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
100003/CS622T.1-PO2	**HIGH**	Identify technically and economically feasible problems by analysing complex engineering problems reaching substantiated conclusions using first principles of mathematics.
100003/CS622T.1-PO3	**HIGH**	Design solutions for complex engineering problems by identifying technically and economically feasible problems.
100003/CS622T.1-PO4	**HIGH**	Identify technically and economically feasible problems by analysis and interpretation of data.
100003/CS622T.1-PO6	**MEDIUM**	Responsibilities relevant to the professional engineering practice by identifying the problem.
100003/CS622T.1-PO7	**MEDIUM**	Identify technically and economically feasible problems by understanding the impact of the professional engineering solutions.
100003/CS622T.1-PO8	**HIGH**	Apply ethical principles and commit to professional ethics to identify technically and economically feasible problems.
100003/CS622T.1-PO9	**MEDIUM**	Identify technically and economically feasible problems by working as a team.
100003/CS622T.1-PO10	**MEDIUM**	Communicate effectively with the engineering community by identifying technically and economically feasible problems.
100003/CS622T.1-P011	**MEDIUM**	Demonstrate knowledge and understanding of engineering and management principles by selecting the technically and economically feasible problems.
100003/CS622T.1-PO12	**HIGH**	Identify technically and economically feasible problems for long term learning.
100003/CS622T.1-PSO1	**MEDIUM**	Ability to identify, analyze and design solutions to identify technically and economically feasible problems.
100003/CS622T.1-PSO2	**MEDIUM**	By designing algorithms and applying standard practices in software project development and Identifying technically and economically feasible problems.
100003/CS622T.1-PSO3	**MEDIUM**	Fundamentals of computer science in competitive research can be applied to Identify technically and economically feasible problems.
100003/CS622T.2-PO1	**HIGH**	Identify and survey the relevant by applying the knowledge of mathematics, science, engineering fundamentals.

100003/CS622T.2-PO2	**HIGH**	Identify, formulate, review research literature, and analyze complex engineering problems get familiarized with software development processes.
100003/CS622T.2-PO3	**HIGH**	Design solutions for complex engineering problems and design based on the relevant literature.
100003/CS622T.2-PO4	**HIGH**	Use research-based knowledge including design of experiments based on relevant literature.
100003/CS622T.2-PO5	**HIGH**	Identify and survey the relevant literature for getting exposed to related solutions and get familiarized with software development processes by using modern tools.
100003/CS622T.2-PO6	**MEDIUM**	Create, select, and apply appropriate techniques, resources, by identifying and surveying the relevant literature.
100003/CS622T.2-PO8	**HIGH**	Apply ethical principles and commit to professional ethics based on the relevant literature.
100003/CS622T.2-PO9	**MEDIUM**	Identify and survey the relevant literature as a team.
100003/CS622T.2-PO10	**HIGH**	Identify and survey the relevant literature for a good communication to the engineering fraternity.
100003/CS622T.2-PO11	**MEDIUM**	Identify and survey the relevant literature to demonstrate knowledge and understanding of engineering and management principles.
100003/CS622T.2-PO12	**HIGH**	Identify and survey the relevant literature for independent and lifelong learning.
100003/CS622T.2-PSO1	**MEDIUM**	Design solutions for complex engineering problems by Identifying and survey the relevant literature.
100003/CS622T.2-PSO2	**MEDIUM**	Identify and survey the relevant literature for acquiring programming efficiency by designing algorithms and applying standard practices.
100003/CS622T.2-PSO3	**MEDIUM**	Identify and survey the relevant literature to apply the fundamentals of computer science in competitive research.
100003/CS622T.3-PO1	**HIGH**	Perform requirement analysis, identify design methodologies by using modern tools & advanced programming techniques and by applying the knowledge of mathematics, science, engineering fundamentals.
100003/CS622T.3-PO2	**HIGH**	Identify, formulate, review research literature for requirement analysis, identify design methodologies and develop adaptable & reusable solutions.

100003/CS622T.3-PO3	**HIGH**	Design solutions for complex engineering problems and perform requirement analysis, identify design methodologies.
100003/CS622T.3-PO4	**HIGH**	Use research-based knowledge including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
100003/CS622T.3-PO5	**HIGH**	Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools.
100003/CS622T.3-PO6	**MEDIUM**	Perform requirement analysis, identify design methodologies and assess societal, health, safety, legal, and cultural issues.
100003/CS622T.3-PO7	**MEDIUM**	Understand the impact of the professional engineering solutions in societal and environmental contexts and Perform requirement analysis, identify design methodologies and develop adaptable & reusable solutions.
100003/CS622T.3-PO8	**HIGH**	Perform requirement analysis, identify design methodologies and develop adaptable & reusable solutions by applying ethical principles and commit to professional ethics.
100003/CS622T.3-PO9	**MEDIUM**	Function effectively as an individual, and as a member or leader in teams, and in multidisciplinary settings.
100003/CS622T.3-PO10	**MEDIUM**	Communicate effectively with the engineering community and with society at large to perform requirement analysis, identify design methodologies.
100003/CS622T.3-PO11	**MEDIUM**	Demonstrate knowledge and understanding of engineering requirement analysis by identifying design methodologies.
100003/CS622T.3-PO12	**HIGH**	Recognize the need for, and have the preparation and ability to engage in independent and lifelong learning in the broadest context of technological change by analysis, identify design methodologies and develop adaptable & reusable solutions.
100003/CS622T.3-PSO3	**MEDIUM**	The ability to apply the fundamentals of computer science in competitive research and prior to that perform requirement analysis, identify design methodologies.
100003/CS622T.4-PO1	**MEDIUM**	Prepare technical report and deliver presentation by applying the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
100003/CS622T.4-PO2	**HIGH**	Identify, formulate, review research literature, and analyze complex engineering problems by preparing technical report and deliver presentation.

100003/CS622T.4-PO3	**MEDIUM**	Prepare Design solutions for complex engineering problems and create technical report and deliver presentation.
100003/CS622T.4-PO4	**MEDIUM**	Use research-based knowledge including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions and prepare technical report and deliver presentation.
100003/CS622T.4-PO5	**MEDIUM**	Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools and Prepare technical report and deliver presentation.
100003/CS622T.4-PO8	**HIGH**	Prepare technical report and deliver presentation by applying ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
100003/CS622T.4-PO9	**HIGH**	Prepare technical report and deliver presentation effectively as an individual, and as a member or leader in teams, and in multidisciplinary settings.
100003/CS622T.4-PO10	**HIGH**	Communicate effectively with the engineering community and with society at large by prepare technical report and deliver presentation.
100003/CS622T.4-PO11	**MEDIUM**	Demonstrate knowledge and understanding of engineering and management principles and apply these to one's own work by prepare technical report and deliver presentation.
100003/CS622T.4-PO12	**HIGH**	Recognize the need for, and have the preparation and ability to engage in independent and lifelong learning in the broadest context of technological change by prepare technical report and deliver presentation.
100003/CS622T.4-PSO1	**MEDIUM**	Prepare a technical report and deliver presentation to identify, analyze and design solutions for complex engineering problems in multidisciplinary areas.
100003/CS622T.4-PSO2	**MEDIUM**	To acquire programming efficiency by designing algorithms and applying standard practices in software project development and to prepare technical report and deliver presentation.
100003/CS622T.4-PSO3	**MEDIUM**	To apply the fundamentals of computer science in competitive research and to develop innovative products to meet the societal needs by preparing technical report and deliver presentation.
100003/CS622T.5-PO1	**HIGH**	Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
100003/CS622T.5-PO2	**HIGH**	Identify, formulate, review research literature, and analyze complex engineering problems by applying engineering and management principles to achieve the goal of the project.

100003/CS622T.5-PO3	**HIGH**	Apply engineering and management principles to achieve the goal of the project and to design solutions for complex engineering problems and design system components or processes that meet the specified needs.
100003/CS622T.5-PO4	**MEDIUM**	Apply engineering and management principles to achieve the goal of the project and use research-based knowledge including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
100003/CS622T.5-PO5	**MEDIUM**	Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools and to apply engineering and management principles to achieve the goal of the project.
100003/CS622T.5-PO6	**MEDIUM**	Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal, and cultural issues and the consequent responsibilities by applying engineering and management principles to achieve the goal of the project.
100003/CS622T.5-PO7	**MEDIUM**	Understand the impact of the professional engineering solutions in societal and environmental contexts, and apply engineering and management principles to achieve the goal of the project.
100003/CS622T.5-PO8	**HIGH**	Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice and to use the engineering and management principles to achieve the goal of the project.
100003/CS622T.5-PO9	**MEDIUM**	Function effectively as an individual, and as a member or leader in teams, and in multidisciplinary settings and to apply engineering and management principles to achieve the goal of the project.
100003/CS622T.5-PO11	**MEDIUM**	Demonstrate knowledge and understanding of engineering and management principles and apply these to one's own work, as a member and leader in a team. Manage projects in multidisciplinary environments and to apply engineering and management principles to achieve the goal of the project.
100003/CS622T.5-PO12	**HIGH**	Recognize the need for, and have the preparation and ability to engage in independent and lifelong learning in the broadest context of technological change and to apply engineering and management principles to achieve the goal of the project.
100003/CS622T.5-PSO1	**MEDIUM**	The ability to identify, analyze and design solutions for complex engineering problems in multidisciplinary areas. Apply engineering and management principles to achieve the goal of the project.

100003/CS622T.5-PSO2	**MEDIUM**	The ability to acquire programming efficiency by designing algorithms and applying standard practices in software project development to deliver quality software products meeting the demands of the industry and to apply engineering and management principles to achieve the goal of the project.
100003/CS622T.5-PSO3	**MEDIUM**	The ability to apply the fundamentals of computer science in competitive research and to develop innovative products to meet the societal needs thereby evolving as an eminent researcher and entrepreneur and apply engineering and management principles to achieve the goal of the project.