

Fall Detection System using Computer Vision

Senior Project

By

**Ali Osseili and Malak Alkazem**

Submitted to the School of Arts & Science of the

Lebanese International University

In part of fulfillment of the requirements for the degree of

**BACHERLOR OF SCIENCE IN**

**COMPUTER SCIENCE AND INFORMATION TECHNOLOGY**

**Supervised by: Dr. Youssef Roumieh**

Spring 2024

# Acknowledgment:

We would like to express our sincere gratitude to the following individuals and entities whose contributions were instrumental in the completion of this senior project:

Dr Youssef Roumieh, PhD., our project advisor, for his invaluable guidance, insightful feedback, and unwavering support throughout the research and development process. His expertise in software development and deployment were crucial in shaping the direction of this project and ensuring its successful outcome

Sarah El-Rashidi, Ms., for providing valuable technical assistance and industry insights during the development phase. Her practical advice and support helped us overcome technical challenges and implement the project effectively.

My collaborator, Malak Alkazem, for her dedication and teamwork throughout the project. Their contributions, including but not limited to a mobile application, and a firebase connection to the system, which were essential in achieving the project goals.

Of course, I can't forget my family. Their unwavering support, like a well-meaning but ultimately distracting cheerleader, ensured this project got the full attention it deserved. While their constant inquiries about "how's it going?" and "is it done yet?" were certainly...motivational, let's just say uninterrupted focus might have shaved off a good eight weeks. But hey, family is family, and who needs deadlines anyway, right? We finished eventually, and that's what counts!

# Abstract:

This project presents a novel approach to fall detection using computer vision for elderly care or assisted living environments. The system leverages a YOLO (You Only Look Once) object detection model, trained on a custom dataset to identify human figures within a video stream. This lightweight model facilitates deployment on resource-constrained devices, making it accessible for a wider range of users.

The real-time video analysis is implemented as an API, enabling integration with a mobile application. Firebase serves as the communication platform, triggering notifications on designated mobile devices whenever a fall event is detected.

This project contributes to improved elder care by providing a non-intrusive, vision-based system for fall detection. The focus on efficient model design and cloud-based communication ensures broader applicability and scalability for real-world scenarios.

Table of Contents

[Acknowledgment: 2](#_Toc166769864)

[Abstract: 3](#_Toc166769865)

[Chapter 1 5](#_Toc166769866)

[1.1 Problem Statement 5](#_Toc166769867)

[1.2 Proposed Solution 5](#_Toc166769868)

[1.3 Technology Constraints 5](#_Toc166769869)

[1.4 Project Objectives 5](#_Toc166769870)

[Chapter 2: Requirements Analysis 6](#_Toc166769871)

[2.1 Functional Requirements 6](#_Toc166769872)

[2.2 Non-Functional Requirements 7](#_Toc166769873)

[2.3 UML Use Cases 7](#_Toc166769874)

[Chapter 3: Application and Design 13](#_Toc166769875)

[Chapter 4: The Model 14](#_Toc166769876)

[4.1 Introduction 14](#_Toc166769877)

[4.2 Data Collection 14](#_Toc166769878)

[4.3 Data Annotation 15](#_Toc166769879)

[4.4 Preprocessing and Augmentation 16](#_Toc166769880)

[4.1 Model Accuracy 16](#_Toc166769881)

# Chapter 1

## 1.1 Problem Statement

Falls are a significant health concern, particularly for the elderly population. According to the Centers for Disease Control and Prevention (CDC), falls are the leading cause of injury deaths among adults aged 65 and older, resulting in over 32,000 fatalities annually [1]. Furthermore, falls can lead to serious injuries like hip fractures, which can significantly impact an individual's mobility and independence.

Traditional fall detection methods often rely on wearable sensors, which can be intrusive, uncomfortable, or forgotten by users. This project addresses the need for a non-intrusive and user-friendly fall detection system suitable for home environments.

## 1.2 Proposed Solution

This project proposes a real-time fall detection system utilizing computer vision and mobile notifications. The system leverages a lightweight YOLO (You Only Look Once) object detection model trained on a custom dataset of human figures. This model can be deployed as an API, enabling integration with a mobile application. Upon detecting a fall event in the video stream, the system triggers a notification sent through Firebase to android mobile devices with the application, alerting caregivers or family members.

## 1.3 Technology Constraints

While computer vision offers a promising approach to fall detection, certain constraints need to be considered:

Accuracy: Accurately distinguishing falls from other activities like sitting or kneeling is crucial.

Computational Cost: Running complex models on resource-constrained mobile devices can be challenging.

Privacy Concerns: Video-based systems raise privacy considerations, requiring careful design and data handling practices.

## 1.4 Project Objectives

This project aims to develop a fall detection system that addresses the aforementioned constraints. The specific objectives include:

Develop a YOLO model with high accuracy in identifying falls within a video stream.

Optimize the model for efficient operation on resource-constrained devices.

Using Roboflow’s API for real-time video analysis, enabling integration with a mobile application.

Utilize Firebase for secure communication and notification delivery when falls are detected.

By achieving these objectives, this project aims to provide a user-friendly and effective fall detection solution for elderly care and assisted living environments.

Reference:

[1] Centers for Disease Control and Prevention (CDC). (2023, March 15). Falls Among Older Adults. Retrieved from <https://www.cdc.gov/falls/index.html>

# Chapter 2: Requirements Analysis

This chapter outlines the functional and non-functional requirements for the fall detection system. Additionally, it details the UML use cases and scenarios to illustrate the system's functionalities from a user perspective.

## 2.1 Functional Requirements

Functional requirements define the specific actions the system should perform. Here are some key functionalities:

* Fall Detection: The core functionality of the system is to analyze the video stream in real-time and detect fall events with high accuracy.
* Emergency Contact Selection: Users can designate emergency contacts to be notified in case of a fall.
* Local Emergency Number: The system automatically dials the local emergency number based on the user's location to connect with emergency services.
* Fall Test: Users can initiate a simulated fall scenario to test the system's detection capabilities and ensure proper operation.
* Response Time Selection: Users can define a delay period before an emergency call is initiated, allowing for potential intervention by a caregiver present at the scene.
* Ringtone Selection: Users can personalize their notification ringtone for fall detection alerts.
* Video Streaming: The system captures video data from the camera and transmits it for processing.
* Notification Delivery: The system sends notifications to designated recipients (mobile app) upon detecting a fall event.
* Emergency Call Cancellation: Residents have the option to cancel an emergency call within the designated response time window.

## 2.2 Non-Functional Requirements

Non-functional requirements define overall system characteristics:

* Fall Detection Accuracy: The system should have a high true positive rate for fall detection, minimizing false positives (alerts for non-fall events).
* Data Security: The system must securely store and transmit user data, including video streams and emergency contact information. Adherence to data privacy regulations is crucial.
* Response Time: There should be minimal delay between fall detection and sending emergency notifications or initiating an emergency call.
* Scalability: The system architecture should allow for adding more users and cameras without impacting performance.
* Ease of Use: The system should be user-friendly for installation, configuration, and operation, catering to users with varying technical skill levels.
* Privacy: The system should have clear and customizable privacy settings. Users should have control over data collection, storage, and usage.

## 2.3 UML Use Cases

UML (Unified Modeling Language) use cases represent system functionalities from the user's perspective. Here are some key use cases for our fall detection system:

|  |  |  |
| --- | --- | --- |
| **Use-Case Name:** | Configure application settings | |
| **Use-Case ID:** | AT-001 | |
| **Priority:** | High | |
| **Primary Actor:** | Resident | |
| **Description:** | This use case describes the scenario where a resident configures the mobile application settings for fall detection. This includes emergency contact information, notification preferences, and response time. | |
| **Precondition:** | The resident has downloaded and installed the fall detection mobile application on their smartphone.  The resident has launched the application and accepted the permissions | |
| **Typical Course of Events:** | **Actor Action** | **System Response** |
| 1. The resident launches the application for the first time.   **If storage permissions are already granted**  The resident taps on the "Settings" menu option within the application.   1. The resident chooses one of the following:   **Allow:** The resident grants storage permission to the application.  **Deny**: The resident denies storage permission to the application.   1. The resident taps on the "Settings" menu option within the application. 2. The resident taps on the "Emergency Contacts" section. 3. The resident Add a new emergency contact by selecting him from contacts list opened. 4. The resident taps on the "Notification Preferences" section. 5. The resident selects their preferred notification settings. 6. The resident taps on the "Fall Response Time" section. 7. The resident chooses their desired fall response time from the available options (e.g., 10 seconds, 30 seconds, etc.). 8. The resident taps on the "Save" button to confirm all configuration changes. | 1. The application checks for necessary storage permissions (contacts and ringtones).   **If storage permissions are already granted**  The application displays the settings screen with various configuration options (Step #2 from original use case).  **If storage permissions are not granted:**   * The application displays a notification requesting permission to access storage for storing emergency contacts and ringtones. * The notification provides options for "Allow" and "Deny".  1. If Allowed: The application hides the permission notification and proceeds to display the settings screen   If Denied: The application deny the resident to enter settings page and configure settings   1. The application displays the settings screen with various configuration options. 2. The application displays a list of existing contacts saved on phone or a blank form to add new contacts. 3. The application saves the contacts accordingly. 4. The application displays options for customizing notifications upon fall detection, such as enabling/disabling ringtones, vibration alerts, and on-screen messages. 5. The application reflects the selected notification preferences. 6. The application displays options to set a delay before initiating a call to emergency contacts after a fall is detected. This allows the resident time to cancel the alert if it's a false positive. 7. The application saves the selected fall response time. 8. The application confirms the settings have been saved successfully. |
| **Postconditions:** | * If the resident grants storage permission, their emergency contact information, notification preferences, and fall response time are configured according to their choices within the application. * If the resident denies storage permission, the application might have limited functionality or return to the main screen. | |

2)

|  |  |  |
| --- | --- | --- |
| **Use-Case Name:** | Resident Triggers Fall Test | |
| **Use-Case ID:** | AT-012 | |
| **Priority:** | Medium | |
| **Primary Actor:** | Resident | |
| **Description:** | This use case describes the scenario where a resident intentionally triggers a fall test using the mobile application. This allows them to verify the fall detection functionality and emergency response process. | |
| **Precondition:** | * The resident has downloaded and installed the fall detection mobile application on their smartphone. * The resident has launched the application and configured their settings (including emergency contacts). | |
| **Typical Course of Events:** | **Actor Action** | **System Response** |
| 1. The resident taps on the dedicated "Fall Test" button within the application. 2. The resident chooses one of the following:   **Yes:** The resident confirms they want to proceed with the fall test  **No:** The resident decides not to perform the fall test at this time.   1. The resident, during the countdown timer, can tap on a "Cancel Test" button displayed within the application. 2. If the countdown timer reaches zero and the resident hasn't canceled the test | 1. The application displays a confirmation dialog box with a clear message like "Are you sure you want to initiate a fall test? 2. **If pressed NO**:   The application closes the confirmation dialog box, and the resident remains on the main screen. No test or notifications are initiated.  **If Yes:**  The application displays a countdown timer indicating the fall response time (as configured in the settings).  An SMS notification is sent to the emergency contacts informing them of a potential fall (clearly stating it's a test).  The selected alert activated on resident’s mobile   1. The application cancels the simulated fall alert.   The countdown timer stops.  No emergency call is initiated   1. The application automatically initiates a phone call to the primary emergency contact. |
| **Postconditions:** | If the resident confirms the fall test, one SMS notification is sent to emergency contacts, and potentially one emergency call is attempted (depending on if the resident cancels before the timer ends).  If the resident cancels the test, no notifications or calls are initiated. | |

3)

|  |  |  |
| --- | --- | --- |
| **Use-Case Name:** | Resident Reads Instructions | |
| **Use-Case ID:** | AT-114 | |
| **Priority:** | Low | |
| **Primary Actor:** | Resident | |
| **Description:** | This use case describes the scenario where a resident accesses and reviews the instructions or tutorials within the fall detection mobile application. This can include information about the application's functionalities, proper usage, and potential limitations. | |
| **Precondition:** | * The resident has downloaded and installed the fall detection mobile application on their smartphone. * The resident has launched the application. | |
| **Typical Course of Events:** | **Actor Action** | **System Response** |
| 1. The resident taps on a dedicated "Instructions" or "Help" button within the application. 2. The resident reads through the instructions, focusing on topics that interest them, such as:   How to configure emergency contact information and notification preferences.  Understanding the fall detection process and response time.  Troubleshooting common issues or limitations of the application.   1. The resident might choose to navigate through different sections of the instructions using provided menus or buttons. 2. The resident taps on a "Close" or "Back" button when they have finished reviewing the instructions. | The application displays a screen or opens a document containing detailed instructions on various aspects of the fall detection system. This information could be presented in different formats like text, images, or even short instructional videos.  The application closes the instructions screen and returns the resident to the previous screen they were on. |
| **Postconditions:** | The resident gains a better understanding of the fall detection application's functionalities and proper usage through the provided instructions. | |

4)

|  |  |  |
| --- | --- | --- |
| **Use-Case Name:** | Emergency Contact Receives Fall Alert | |
| **Use-Case ID:** | AT-005 | |
| **Priority:** | High | |
| **Primary Actor:** | Emergency Contact | |
| **Description:** | This use case describes the scenario where a designated emergency contact receives a notification and potentially a phone call due to a fall detection event triggered by the resident. | |
| **Precondition:** | * The resident has downloaded and installed the fall detection mobile application on their smartphone. * The resident has configured the application settings, including adding the emergency contact's phone number. * A fall event is detected by the Model | |
| **Typical Course of Events:** | . The fall detection model notifies the application on the resident's phone and registers a possible fall.    The emergency contact receives an SMS notification from the fall detection application.  Upon receiving the notification or phone call, the emergency contact takes necessary actions based on the situation, such as:  Calling the resident's phone number to check on their well-being.  Dispatching help to the resident's location (if the location information is provided).  Contacting other family members or emergency services depending on the urgency of the situation. | 1)The application initiates its emergency response protocol, which might include:  Sending an SMS notification to the emergency contacts listed in the app's settings.  Initiating an automated phone call to the primary emergency contact (depending on application configuration).  2)The notification clearly informs the contact about a potential fall incident involving the resident. It might include details like:  A message stating, "Fall Alert: [Resident's Name] may have fallen."  The resident's location information (if enabled and permitted by the resident's phone settings).  A link to a map app showing the resident's location (optional). |
| **Postconditions:** | The emergency contact is alerted about a potential fall incident involving the resident, allowing them to take appropriate action to ensure the resident's safety. | |

# Chapter 3: Application and Design

This chapter describes the system design, including the mobile application, the model architecture and the terminal setup for the model.

# Chapter 4: The Model

This chapter will detail the development and implementation of a custom You Only Look Once (YOLO) model for fall detection in computer vision.

## 4.1 Introduction

You Only Look Once (YOLO) is a powerful object detection framework known for its speed and accuracy. In our fall detection project, we leveraged a custom YOLO model. This model is trained to analyze entire images at once, allowing it to efficiently identify a person's pose and determine if they have fallen within the frame. This real-time detection capability is crucial for our project, as it enables us to quickly react to potential fall situations.

## 4.2 Data Collection

Building a robust fall detection model hinges on a diverse and informative dataset. To achieve this, we weren't limited to a single source. We collected images from:

* YouTube videos
* Instagram posts
* Roboflow

Additionally, we explored other image platforms that offered relevant content. This multi-source approach ensured our model encountered a wide range of fall scenarios, backgrounds, and lighting conditions, ultimately enhancing its ability to generalize and perform accurately in real-world situations.

We collected a total of 2400 images of people in falling motion.

**A significant amount of effort was invested in this step, as the quality of the data, directly influences the model’s accuracy.**

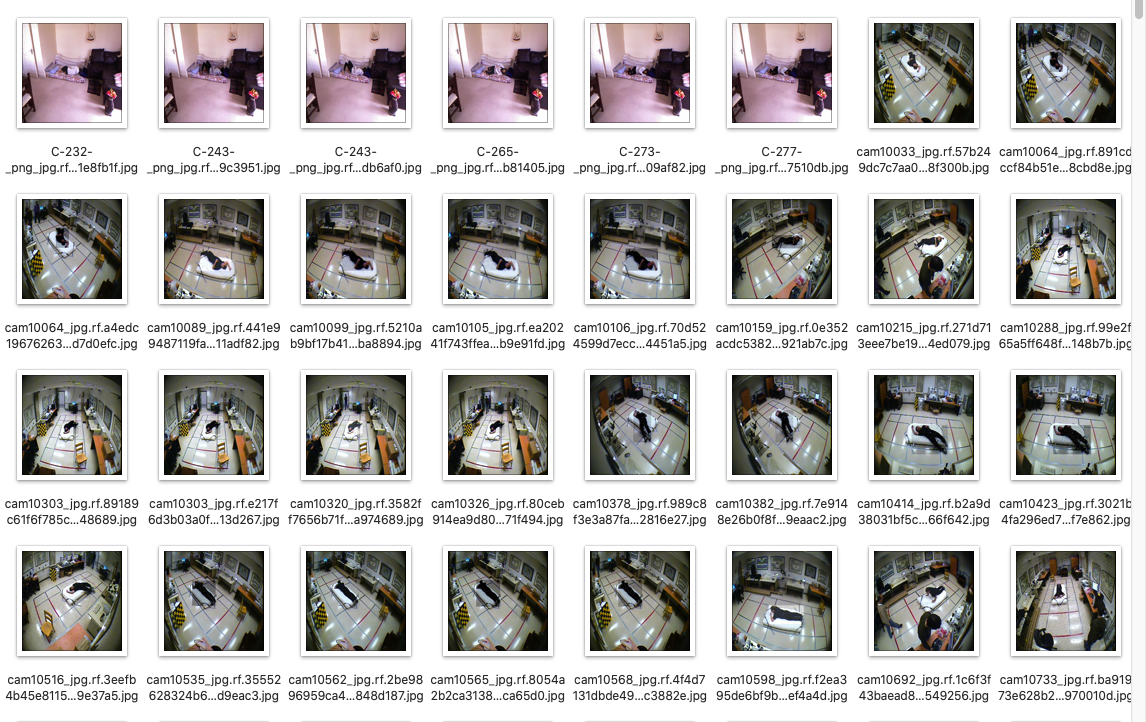


Figure 1: A screenshot showcasing a part of the dataset that was used to train the model

## 4.3 Data Annotation

After collecting the raw images, the next critical step involved meticulous annotation. This process entailed manually labeling each image to identify the presence or absence of a fall. We meticulously defined bounding boxes around individuals in the images, particularly focusing on poses indicative of falls.

This precise annotation provided the ground truth data for our YOLO model, allowing it to learn the visual characteristics associated with fall events. The quality and comprehensiveness of this annotation process directly influence the model's ability to accurately detect falls in unseen situations.

We used Roboflow to annotate the images.

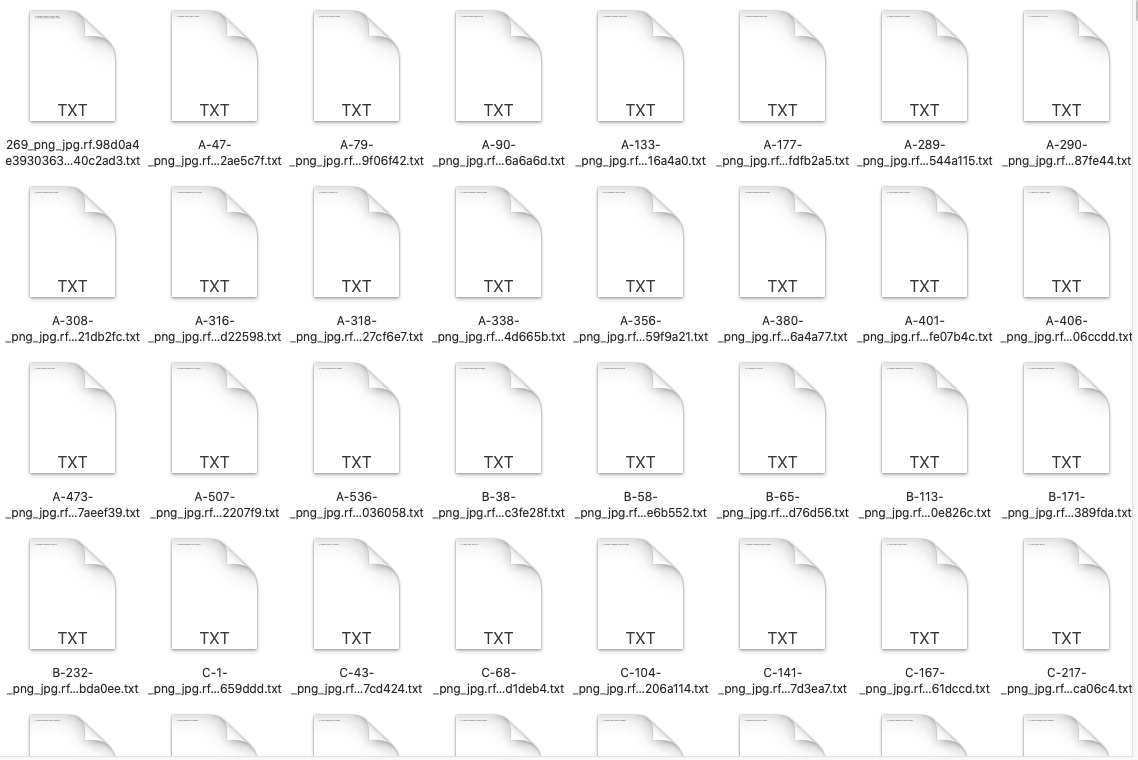


Figure 2: A screenshot showing the image annotation data

## 4.4 Preprocessing and Augmentation

Prior to training our YOLO model, we implemented a data pre-processing pipeline. This involved essential steps like resizing images to a consistent size and normalizing pixel values for better training efficiency. Initially, we experimented with data augmentation techniques like image rotation and flipping. However, during development of the first two model versions, we encountered a significant challenge. Augmenting images of fallen people by rotating or flipping them inadvertently caused them to resemble images of people standing. This confusion led to a high number of false positives in our model. To address this issue, the final version of the YOLO model relied minimally on augmentation and focused primarily on resizing and normalization. This approach minimized the risk of introducing misleading variations during training and ultimately led to improved fall detection accuracy.

## 4.5 Accuracy and Metrics

Our custom YOLO model achieved impressive results in fall detection. The model's mean Average Precision (mAP) reached 91.4%, demonstrating its overall effectiveness in correctly identifying falls across various scenarios. Furthermore, it achieved a high precision of 96.8%, indicating a low rate of false positives, meaning the model rarely flagged non-fall events as falls. While recall reached 81.4%, signifying the model successfully detected most actual falls in the test data. It's important to note that during our own internal testing with unseen data featuring different lighting and angles, the model remarkably identified falls in all 10 test cases. This exceptional performance in a controlled setting highlights the model's potential for real-world application.

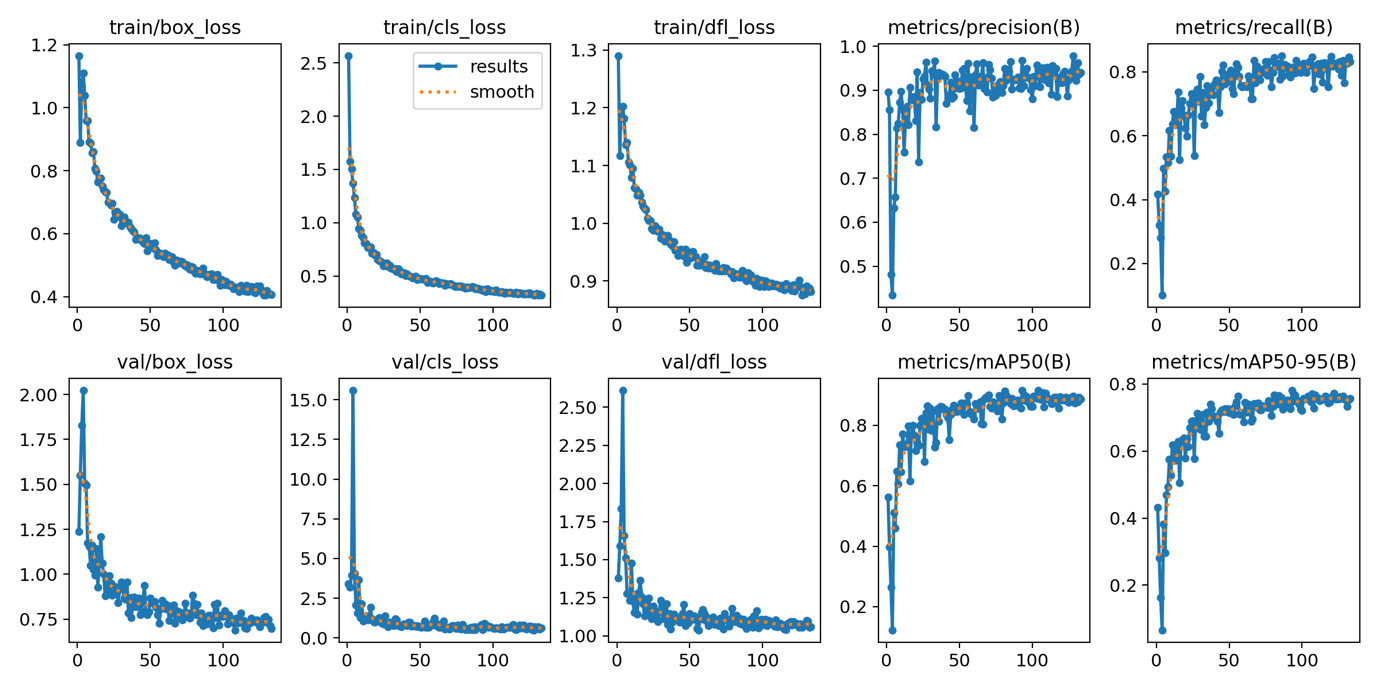


Figure 3: An image showing the metrics of our final model