**Military Target and Equipment detection using Computer Vision**

**ABSTRACT**

Military target detection technology is the foundation and key to perceive and analyze the battlefield situation, and it is also the premise of target tracking technology. Aiming at the task of military target detection, the detection performance of traditional detection algorithms is poor in complex environment. We realized automatic detection of military targets in complex environment through deep learning. In this project, we improved the components of YOLOv3 and proposed a novel military target detection algorithm (YOLO-v7). We have built a military target dataset composed of armed men with different weapons, which provides a test environment for various object detection algorithms.

The problem at hand is the need for an advanced and precise military target and equipment detection system that overcomes the limitations of existing solutions. Current YOLOv7-based systems, while promising, exhibit issues such as false positives, dataset limitations, struggles with complex backgrounds, user interface complexity, and demanding real-time processing requirements.

Consequently, there is a critical need for the development of a comprehensive system that addresses these challenges, ensuring high precision, adaptability to diverse scenarios, user-friendliness, and efficient real-time processing. This system should empower military and security personnel with an accessible and dependable tool for rapid and accurate target identification in dynamic and cluttered environments, ultimately enhancing operational efficiency, situational awareness, and risk mitigation in defense and security operations.

**CHAPTER 1**

**INTRODUCTION**

**1.1 Project Overview**

Military target detection technology is the foundation and key to perceive and analyze the battlefield situation, and it is also the premise of target tracking technology. Aiming at the task of military target detection, the detection performance of traditional detection algorithms is poor in complex environment. We realized automatic detection of military targets in complex environment through deep learning. In this project, we improved the components of YOLOv3 and proposed a novel military target detection algorithm (YOLO-v7). We have built a military target dataset composed of armed men with different weapons, which provides a test environment for various object detection algorithms.

The problem at hand is the need for an advanced and precise military target and equipment detection system that overcomes the limitations of existing solutions. Current YOLOv7-based systems, while promising, exhibit issues such as false positives, dataset limitations, struggles with complex backgrounds, user interface complexity, and demanding real-time processing requirements.

Consequently, there is a critical need for the development of a comprehensive system that addresses these challenges, ensuring high precision, adaptability to diverse scenarios, user-friendliness, and efficient real-time processing. This system should empower military and security personnel with an accessible and dependable tool for rapid and accurate target identification in dynamic and cluttered environments, ultimately enhancing operational efficiency, situational awareness, and risk mitigation in defense and security operations.

**1.2 Motivation**

This project is motivated by the imperative need to modernize and optimize military and security operations. In an era where operational efficiency is paramount, manual target identification methods have proven to be slow and prone to errors. The aim is to leverage advanced computer vision technologies, including YOLOv7, Pytorch, and OpenCV, to expedite the identification process, enhance situational awareness, and mitigate risks.

By automating target detection, this project seeks to optimize resource allocation, reduce the potential for friendly fire incidents, and capitalize on the latest technological advancements. Ultimately, this project aligns with the broader trend of integrating artificial intelligence into defence and security, striving to enhance operational capabilities and safety

**CHAPTER 2**

**LITERATURE SURVEY**

**Paper 1:**

**Title: Computer Vision-Based Military Tank Recognition Using Object Detection Technique: An application of the YOLO Framework**

**Authors:** Sikandar Ali , Abdullah, Ali Athar, Maisam Ali, Ali Hussain, Hee-Cheol Kim

**Summary:** Military object detection is crucial for defense systems. It involves tracking, securing, and surveilling territories. These systems must be highly efficient, reliable, and accurate, as even small errors can lead to significant destruction and loss. Therefore, real-time automatic object detection is essential in today's world.

This YOLOv5 model successfully identifies military tanks and flags with high confidence and precision. The authors conducted experiments involving YOLOv3, YOLOv4, and four YOLOv5 versions (YOLOVv5s, YOLOv5m, YOLOV5l, YOLOV5xl) using a dataset of 922 tank and flag images. The dataset was split into 80% for training, 10% for validation, and 10% for testing.

Overall, YOLOv5, being one of the latest and fastest real-time object detection methods, has the potential to enhance military surveillance systems. It enables military personnel to respond promptly and proactively to potential threats.

**Paper 2:**

**Title: Military Target Detection Method Based on Improved YOLOv5**

**Authors:** Xiuli Du, Linkai Song, Yana Lv and Xutong Qin

**Summary:** This paper addresses the challenge of detecting military targets with limited hardware resources for weapons platforms. The proposed method prioritizes network efficiency, mean average precision (mAP), and detection speed. It builds upon the YOLOv5 algorithm.

Firstly, the paper introduces the use of the Stem block module instead of the Focus module, enhancing feature expression and reducing network model parameters and computational load. Secondly, it designs a MobileNetV2-Convolutional Block Attention Module (MNtV2-CBAM) structure, integrating MobileNetV2 with CBAM mechanisms. This design decreases network parameters and computation while improving detection performance.

The experimental results demonstrate that compared to YOLOv5, the proposed method increases mAP by 1.3% and significantly reduces parameters (67.45%) and computational load (73.17%). This approach is well-suited for resource-constrained weapon equipment platforms, enhancing military intelligence reconnaissance and analysis capabilities, reducing commander decision-making time, and greatly improving troop combat effectiveness.

**Paper 3:**

**Title: Research on military target detection method based on YOLO method**

**Author:** Huibai Wang and Ji Han

**Summary:** In the realm of military target recognition, addressing issues such as the potential loss of small target detection results and slow detection speed is crucial. To tackle these challenges, this paper introduces the YOLOG2S target recognition detection algorithm, which achieves a reduction in parameters and model size while maintaining respectable accuracy.

In the original YOLO v5 model, the 2nd, 3rd, and 4th C3 modules in the backbone are substituted with C3G2 modules. This substitution effectively reduces the model's size and significantly enhances the detection speed. Additionally, accuracy is fine-tuned by modifying the RELU activation function within the C3G2 module.

The experimental outcomes are promising, demonstrating that YOLO-G2S achieves an impressive average accuracy of 95%. Moreover, it accomplishes a notable reduction of 7.3% in the number of parameters and a substantial 17.8% decrease in computational workload. As a result, this enhanced algorithm proves to be well-suited for deployment in military target recognition scenarios, where efficiency, accuracy, and streamlined model size are of paramount importance.

**Paper 4:**

**Title: An Aircraft Target Detection Method Based on Regional Convolutional Neural Network for Remote Sensing Images**

**Author:** Bing Wang, Yan Zhou, Huainian Zhang and Ning Wang

**Summary:** This paper focuses on the critical task of detecting aircraft targets in battlefield surveillance and reconnaissance using remote sensing images. Detecting aircraft targets serves two primary purposes: swiftly gathering intelligence on enemy military activities and aiding air target identification, as well as assessing the importance of military airfields and analyzing enemy operational intentions for precise strikes.

The current aircraft detection method employs a single convolutional neural network for feature extraction and recognition, but it struggles to accurately capture aircraft characteristics and account for scale differences among various aircraft types, leading to less precise results.

To address this challenge, the paper utilizes deep residual networks to extract aircraft target features and analyzes different aircraft sizes using K-means clustering, defining representative aircraft sizes.

It introduces the Aircraft Targets Region Proposal Network (ATRPN) to combine geometric characteristics of various aircraft. Building on the Faster R-CNN framework, the proposed ATRPN R-CNN remote sensing image aircraft target detection method incorporates deep residual networks and ATRPN for candidate box generation.

The paper establishes a comprehensive aircraft target detection dataset and conducts performance comparison experiments against Faster R-CNN and single network target multi-scale detection framework (SSD). The results demonstrate superior detection accuracy across various scenes and aircraft targets. This method enhances military intelligence capabilities in diverse scenarios**.**

**CHAPTER 3**

**SYSTEM ANALYSIS**

**3.1 Problem Statement**

The problem at hand is the need for an advanced and precise military target and equipment detection system that overcomes the limitations of existing solutions. Current YOLOv7-based systems, while promising, exhibit issues such as false positives, dataset limitations, struggles with complex backgrounds, user interface complexity, and demanding real-time processing requirements.

Consequently, there is a critical need for the development of a comprehensive system that addresses these challenges, ensuring high precision, adaptability to diverse scenarios, user-friendliness, and efficient real-time processing. This system should empower military and security personnel with an accessible and dependable tool for rapid and accurate target identification in dynamic and cluttered environments, ultimately enhancing operational efficiency, situational awareness, and risk mitigation in defense and security operations.

**3.2 Existing System**

Segmentation for target detection in complex background, and applied it to military reconnaissance related fields.

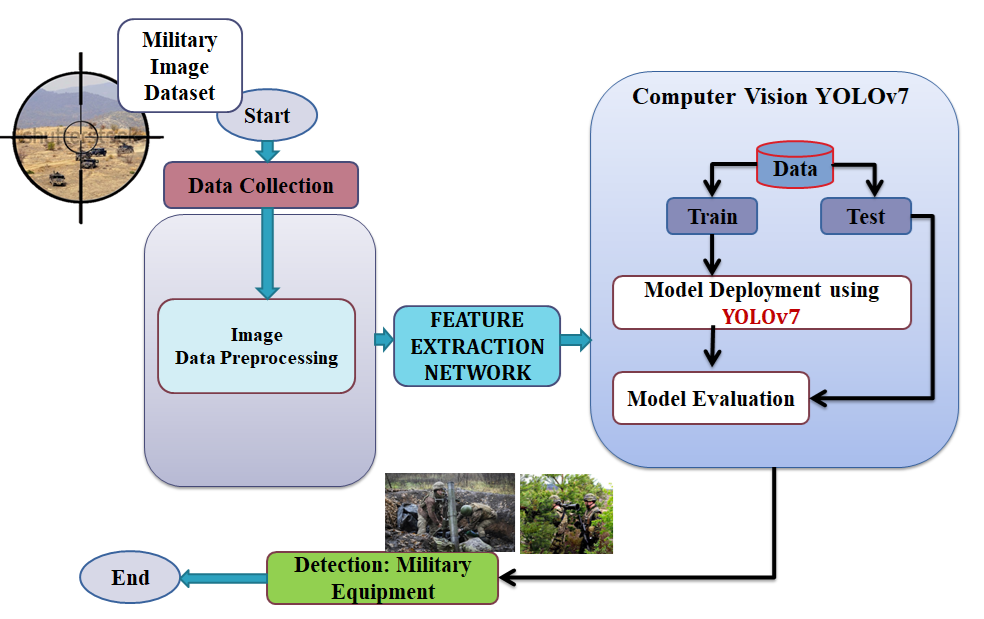
Target detection and classification using fuzzy inference system.

Automatic Target Detection (ATD) algorithm for Charge Coupled Device (CCD) images.

Automatic target recognition in Synthetic Aperture Radar (SAR) aerial images by using neural network

**3.3 Proposed System**

The research methodology for this project entails a systematic approach designed to achieve precise and efficient military target and equipment detection within the ambit of computer vision using the trained Pytorch files for the targets. It commences with the acquisition of a diversified dataset meticulously annotated on Roboflow, encompassing an array of military target scenarios in different angles and lighting conditions. This dataset is subjected to preprocessing, including resizing and class modification, to facilitate subsequent training procedures.



**Figure 3.1 Proposed System**

The YOLOv7 architecture is selected as the cornerstone of object detection, necessitating the strategic partitioning of the dataset into training, validation, and test sets. Through rigorous training, we aim to optimize precision while minimizing false positives. The model's accuracy, precision, recall, and F1 score are assessed using validation and test datasets. This rigorous analysis informs iterative improvements. Simultaneously, we embark on the development of a user-friendly web application, harnessing Flask for real-time detection.

The YoloV7 files and trained Pytorch files along with the Flask User-interface html and webapp python files are run on the selected environment. Real-time webcam, and RTSP stream support are incorporated, while robust security measures ensure data protection. The user can then be able to upload images by browsing through local storage and then upload it for the model to be traced and target to be detected which will be displayed on the output section.

This meticulous methodology aspires to bridge the gap between theoretical computer vision concepts and practical, deployable systems, ultimately empowering military and security personnel with a versatile and reliable tool for enhanced operational effectiveness and situational awareness.

**CHAPTER 4**

**METHODOLOGY**

**4.1 Deep Learning**

Deep learning is a pivotal technology in system analysis, enabling computers to autonomously discover patterns, make predictions, and derive insights from complex data. At its core, deep learning leverages artificial neural networks, which are composed of interconnected layers of artificial neurons, mimicking the human brain's structure.

Deep learning models, particularly convolutional neural networks (CNNs) for image data and recurrent neural networks (RNNs) for sequential data, excel in recognizing intricate patterns and temporal dependencies within system-related data streams. For instance, they can identify defects in manufacturing processes, predict equipment failures based on sensor data, or optimize energy consumption in buildings.

Moreover, deep learning is adaptable and can be fine-tuned for specific system analysis tasks. Its versatility allows it to be applied across diverse industries, from manufacturing and healthcare to finance and infrastructure management.

While deep learning holds immense potential in system analysis, it requires substantial computational resources and often relies on extensive labeled datasets for training. Nevertheless, its capacity to unearth hidden insights within complex data sets makes it an indispensable tool for optimizing and enhancing various systems.

**4.2 Object Detection**

Object detection stands as a pivotal technological cornerstone within the sphere of system analysis. Its primary function is to serve as the visual perception for automated systems, adept at pinpointing and ascertaining specific objects or entities present in images or video streams. The applications of this technology are diverse and wide-ranging, encompassing domains such as surveillance, autonomous vehicles, robotics, and quality control.

At its core, object detection systems harness the potency of sophisticated computer vision algorithms and deep learning models to dissect visual data and draw out meaningful insights. This process hinges on the system's ability to recognize and pinpoint objects of interest within a given frame or scene, typically denoting their location with bounding boxes and furnishing essential details about their class or category.

In the realm of system analysis, object detection performs multifaceted roles. It bolsters security and surveillance measures, granting heightened vigilance and rapid response capabilities. It underpins autonomous systems by facilitating obstacle detection and avoidance. It ensures meticulous quality control in manufacturing processes, elevating the standard of end products. In the retail sphere, it optimizes inventory management, augmenting both efficiency and customer experience. Furthermore, object detection finds a niche in medical diagnosis by meticulously analyzing medical images. It contributes to environmental conservation by monitoring natural habitats and wildlife. Lastly, it revolutionizes agriculture through precision farming practices, ushering in increased yields and resource efficiency.

The triumvirate of accuracy, swiftness, and adaptability displayed by object detection is indispensable for the efficacious analysis and operation of automated systems across an expansive spectrum of domains. By doing so, it contributes significantly to heightened efficiency, safety, and overall productivity.

**4.3 YOLOV7**

YOLOv7 (You Only Look Once version 7) is a state-of-the-art object detection model with significant implications for system analysis. This computer vision system offers a streamlined approach to identifying objects in images and videos, making it particularly valuable in applications where real-time detection is crucial.

The architecture of YOLOv7 revolves around deep convolutional neural networks, allowing it to process images rapidly and accurately in a single pass. This efficiency makes it adaptable to diverse hardware platforms, from embedded systems to high-performance servers. Furthermore, YOLOv7 strikes a balance between speed and precision, making it suitable for various use cases, such as surveillance, autonomous vehicles, and industrial automation.

The model's scalability is a key feature, with multiple model sizes and configurations available to suit different requirements. It can be fine-tuned for specific object detection tasks and is open-source, fostering collaboration and customization within the computer vision community.

System analysis of YOLOv7 involves evaluating its computational demands, accuracy, latency, and ease of integration into the target system. By understanding these aspects, developers and researchers can determine whether YOLOv7 is the right choice for their application, whether it's enhancing security systems, enabling autonomous navigation, or powering other object detection solutions.

**4.4 Flask**

Flask is a lightweight and versatile web framework for Python that plays a valuable role in system analysis by facilitating the development of web-based applications and APIs. It provides a foundation for building robust and scalable systems for data visualization, management, and analysis.

Flask's simplicity and modularity make it well-suited for system analysis tasks. Its micro-framework design means it provides only the essential components for web development, allowing developers to choose and integrate additional libraries and tools as needed. This flexibility makes Flask an excellent choice for customizing web applications tailored to specific system analysis requirements.

In system analysis, Flask can be used to create web-based dashboards and user interfaces for data visualization, enabling stakeholders to interact with and gain insights from complex data sets. It also serves as a platform for building APIs that facilitate data exchange between different components of a system or enable remote access to analytical tools and processes.

Flask's support for extensions and third-party integrations further enhances its capabilities. It can be easily combined with libraries for data manipulation, machine learning, and database access, making it a powerful tool for processing and analyzing data within a system.

Moreover, Flask is well-documented and has a strong community of developers, which means there is a wealth of resources and expertise available for system analysts looking to leverage this framework. Its simplicity, flexibility, and community support make Flask an invaluable asset for building web-based solutions that enhance system analysis capabilities and facilitate data-driven decision-making.

**4.5 OpenCV:**

OpenCV (Open-Source Computer Vision Library) is an open-source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. Being an Apache 2 licensed product, OpenCV makes it easy for businesses to utilize and modify the code.

The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc. OpenCV has more than 47 thousand people of user community and estimated number of downloads exceeding 18 million. The library is used extensively in companies, research groups and by governmental bodies.

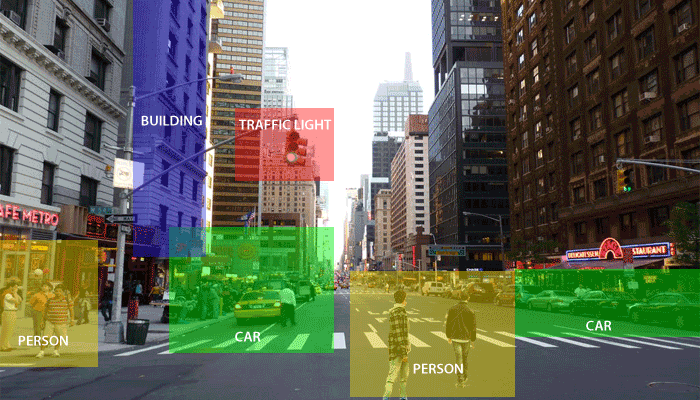


Figure : Purpose of CV

Along with well-established companies like Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda, Toyota that employ the library, there are many startups such as Applied Minds, VideoSurf, and Zeit era, that make extensive use of OpenCV. OpenCV’s deployed uses span the range from stitching Streetview images together, detecting intrusions in surveillance video in Israel, monitoring mine equipment in China, helping robots navigate and pick up objects at Willow Garage, detection of swimming pool drowning accidents in Europe, running interactive art in Spain and New York, checking runways for debris in Turkey, inspecting labels on products in factories around the world on to rapid face detection in Japan.

It has C++, Python, Java and MATLAB interfaces and supports Windows, Linux, Android and Mac OS. OpenCV leans mostly towards real-time vision applications and takes advantage of MMX and SSE instructions when available. A full-featured CUDA and OpenCL interfaces are being actively developed right now. There are over 500 algorithms and about 10 times as many functions that compose or support those algorithms. OpenCV is written natively in C++ and has a templated interface that works seamlessly with STL containers.

### 4.5.1 OpenCV Functionality

### Image/video I/O, processing, display (core, imgproc, high Gui)

### Object/feature detection (objdetect, features2d, nonfree)

### Geometry-based monocular or stereo computer vision (calib3d, stitching, videostab)

### Computational photography (photo, video, superres)

### Machine learning & clustering (ml, flann)

### CUDA acceleration (gpu)

**4.5.2 Features of OpenCV Library**

* Read and write images.
* Capture and save videos.
* Process images (filter, transform)
* Perform feature detection.
* Detect specific objects such as faces, eyes, cars, in the videos or images.
* Analyze the video, i.e., estimate the motion in it, subtract the background, and track objects in it.

### The purpose of computer vision is to understand the content of the images. It extracts the description from the pictures, which may be an object, a text description, and three-dimension model, and so on. For example, cars can be facilitated with computer vision, which will be able to identify and different objects around the road, such as traffic lights, pedestrians, traffic signs, and so on, and acts accordingly.

Computer vision allows the computer to perform the same kind of tasks as humans with the same efficiency. There are a two main task which are defined below:

* Object Classification - In the object classification, we train a model on a dataset of particular objects, and the model classifies new objects as belonging to one or more of your training categories.
* Object Identification - In the object identification, our model will identify a particular instance of an object - for example, parsing two faces in an image and tagging one as Virat Kohli and other one as Rohit Sharma.

**5.6 Python**

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. It’s high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

Since there is no compilation step, the edit-test-debug cycle is incredibly fast. Debugging Python programs is easy: a bug or bad input will never cause a segmentation fault. Instead, when the interpreter discovers an error, it raises an exception. When the program doesn't catch the exception, the interpreter prints a stack trace. A source level debugger allows inspection of local and global variables, evaluation of arbitrary expressions, setting breakpoints, stepping through the code a line at a time, and so on. The debugger is written in Python itself, testifying to Python's introspective power. On the other hand, often the quickest way to debug a program is to add a few print statements to the source: the fast edit-test-debug cycle makes this simple approach very effective. It’s easy to learn syntax and portability capability makes it popular these days.

Python is used for server-side web development, software development, mathematics, and system scripting, and is popular for Rapid Application Development and as a scripting or glue language to tie existing components because of its high-level, built-in data structures, dynamic typing, and dynamic binding. Program maintenance costs are reduced with Python due to the easily learned syntax and emphasis on readability. Additionally, Python's support of modules and packages facilitates modular programs and reuse of code. Python is an open source community language, so numerous independent programmers are continually building libraries and functionality for it.

**The followings facts give us the introduction to Python:**

1. Python was developed by Guido van Rossum at Stichting Mathematisch Centrum in the Netherlands.

2. It was written as the successor of programming language named ‘ABC’.

3. It’s first version was released in 1991.

4. The name Python was picked by Guido van Rossum from a TV show named Monty Python’s Flying Circus.

5. It is an open source programming language which means that we can freely download it and use it to develop programs. It can be downloaded from www.python.org.

6. Python programming language is having the features of Java and C both. It is having the elegant ‘C’ code and on the other hand, it is having classes and objects like Java for object-oriented programming.

7. It is an interpreted language, which means the source code of Python program would be first converted into bytecode and then executed by Python virtual machine.

**CHAPTER 5**

**SYSTEM DESIGN**

System Design is the process of designing the architecture and components of a machine learning system to accomplish specific objectives efficiently and effectively. It encompasses decisions about data flow, model architecture, deployment strategies, and the overall structure of the system. critical phase that requires collaboration between data scientists, machine learning engineers, and domain experts.

**5.1 Architecture Design**

An architecture diagram is a visual representation of all the elements that make up part, or all, of a system. Above all, it helps the engineers, designers, stakeholders — and anyone else involved in the project — understand a system or app’s layout.

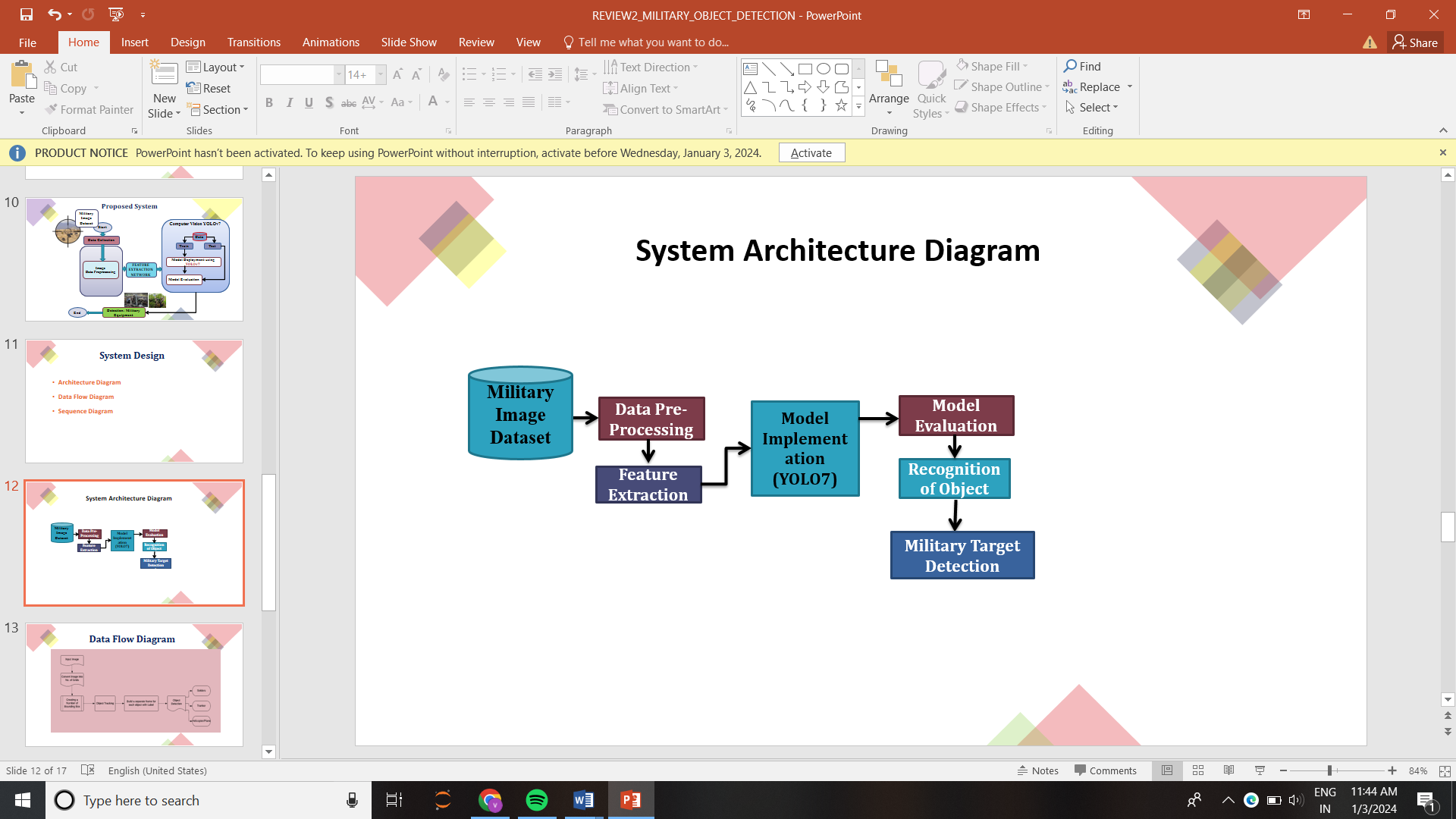


Figure 5.1: Architecture Diagram

**5.2 Data Flow diagram**

This DFD is a conceptual representation, and the actual implementation may involve more detailed processes, interactions, and data flows depending on the specific requirements of the wild animal prediction system and also DFD is a visual representation of how data flows within a system. In the context of a wild animal prediction system. It illustrates how data moves and is processed within a machine learning system. DFDs provide a high-level view of the flow of data, helping to understand the different components and processes involved in a machine learning workflow. They are valuable for both designing and communicating the architecture of a machine learning system.

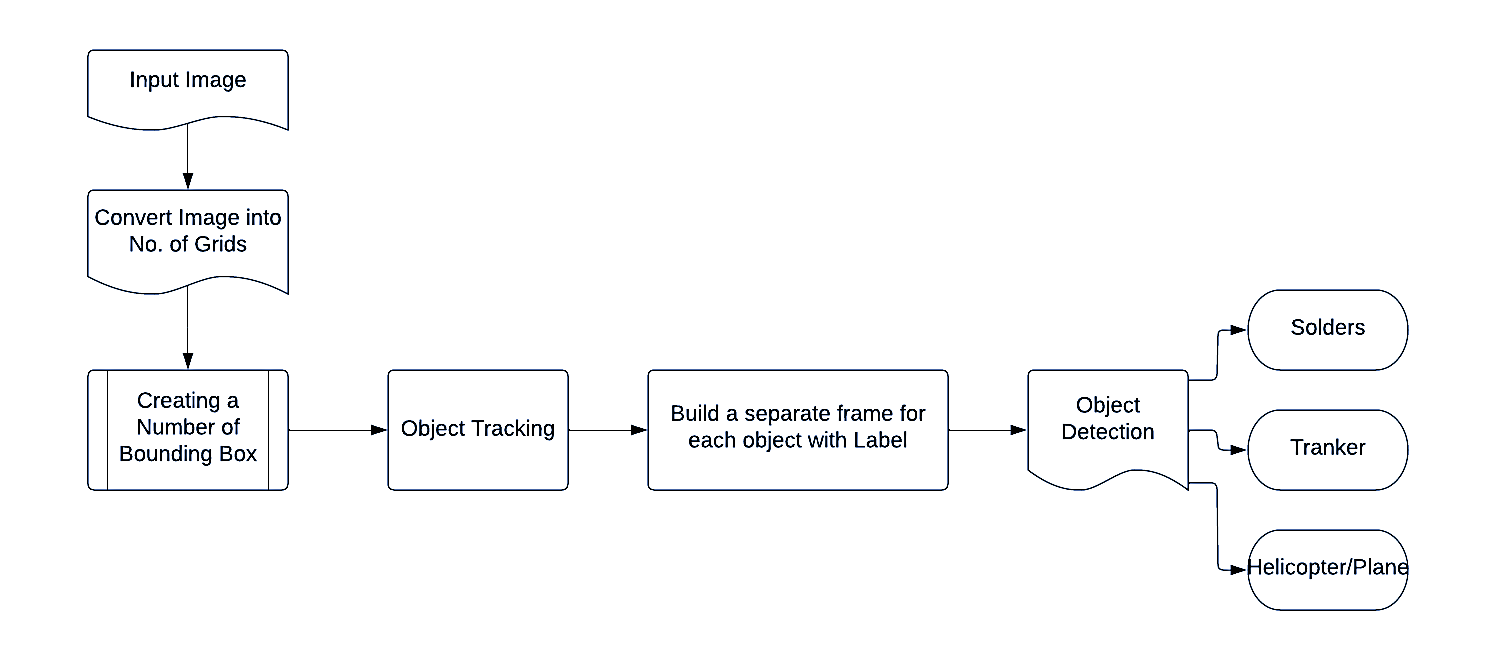


Figure 5.2: Data Flow Diagram

**5.3 Sequence Diagram**

A sequence diagram is a type of interaction diagram that illustrates the interactions between objects or components in a system over time. In the context of a wild animal prediction system using machine learning, a sequence diagram can depict the flow of messages and actions among various components during the prediction process.

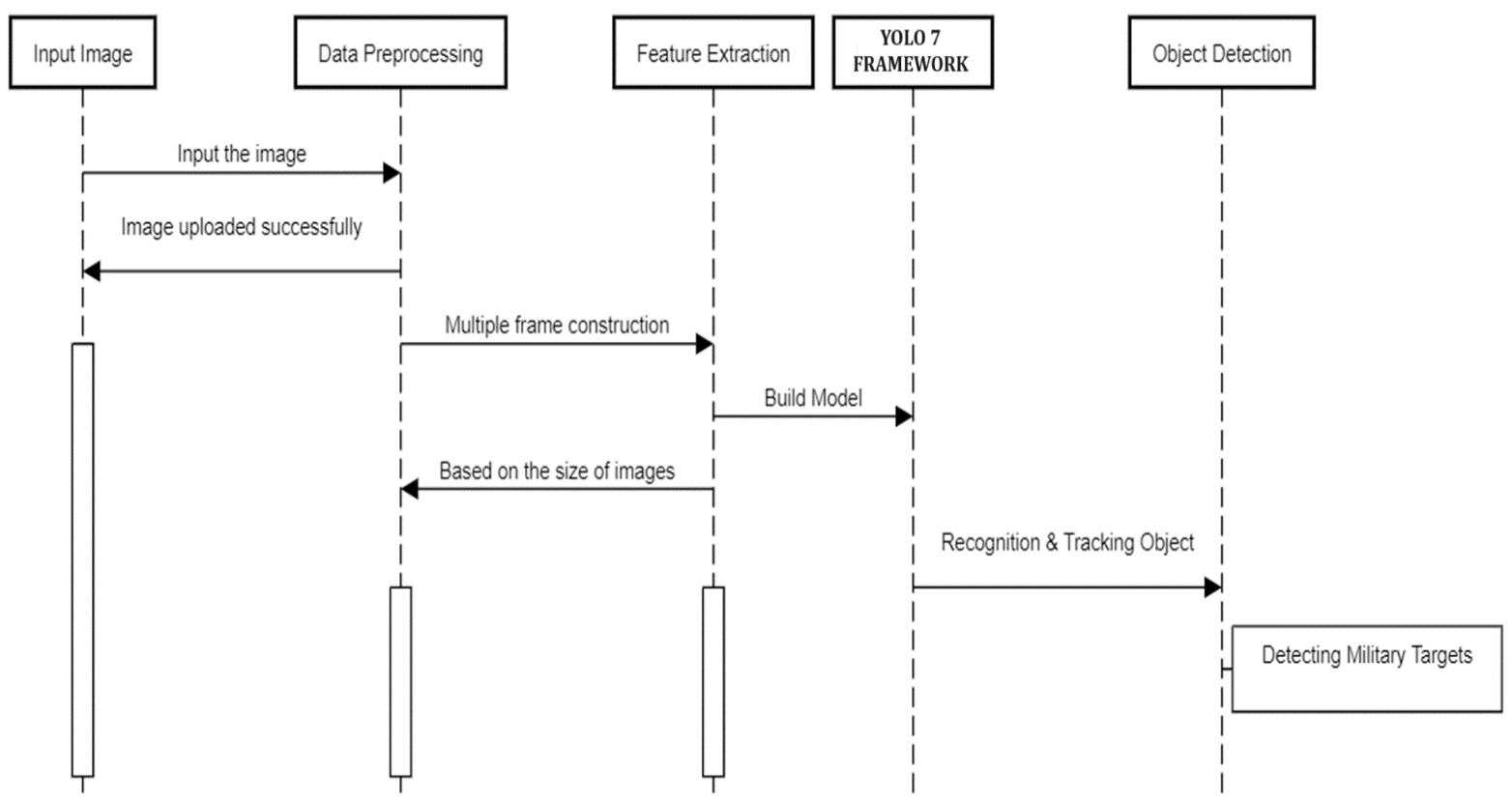
****

Figure 5.3: Sequence Diagram

**CHAPTER 6**

**SYSTEM REQUIREMENTS SPECIFICATIONS**

A Software Requirements Specification (SRS) is a comprehensive description of the intended purpose and environment for software under development. The SRS fully describes what the software will do and how it will be expected to perform. Software requirements specification permits a rigorous assessment of requirements before design can begin and reduces later redesign. It should also provide a realistic basis for estimating product costs, risks, and schedules.

**6.1 HARDWARE REQUIREMENTS**

The following are the hardware requirements:

CPU: A quad-core CPU (4 cores) should be sufficient. More cores might benefit multi-threaded tasks but usually does not affect model training significantly.

GPU: For optimal performance and faster training and inference, we recommend using a GPU with a minimum of 8GB of memory. NVIDIA GPUs with CUDA support is ideal for this purpose.

RAM: At least 16GB of RAM is recommended.

Storage: Enough storage space to store your dataset and trained models.

**6.2 SOFTWARE REQUIREMENTS**

The following are the software requirements:

Operating system: A 64-bit operating system that supports your selected hardware and necessary software (like CUDA for Nvidia GPUs) should work. Common choices include various distributions of Linux, such as Ubuntu.

Python: Python 3.8 or higher

PyTorch: PyTorch 1.8 or higher

CUDA: CUDA 11.6 or higher (required for Nvidia GPUs)

**6.3 FUNCTIONAL REQUIREMENTS**

1. **Image Upload and Processing:** Users should have the capability to upload images for military target and equipment detection, and the system must process uploaded images for real-time object detection.
2. **Object Detection and Bounding Box Visualization:** The system should accurately detect military targets and equipment within uploaded images, and detected objects must be visually highlighted with bounding boxes overlaid on the images.
3. **Class Identification and Confidence Reporting:** The system must display detected object classes and their corresponding confidence scores to enhance situational awareness.
4. **Webcam and RTSP Stream Support:** The system should support real-time object detection from webcam and RTSP feed streams, expanding its utility.
5. **Hardware Compatibility:** The system must be compatible with a range of hardware configurations, including both high-performance and resource-constrained devices.

**6.4 NON-FUNCTIONAL REQUIREMENTS**

1. **Performance:** The application must provide real-time target and equipment detection capabilities, ensuring that military personnel can respond swiftly to threats. Low latency is crucial to minimize delays in critical situations.
2. **Scalability:** The system should be able to scale horizontally to handle sudden surges in usage during military operations or emergencies. Horizontal scalability ensures that additional resources can be added seamlessly to meet increased demand.
3. **Security:** Robust security measures are essential to protect sensitive military data. Data encryption, strict authentication, and authorization controls are vital to prevent unauthorized access and data breaches.
4. **Availability**: High availability is paramount in military applications. The system should be operational 24/7 with minimal downtime for maintenance or updates, ensuring that it's ready to respond to threats at any time.
5. **Resource Utilization:** Efficient use of computing resources is critical to minimize operational costs. Monitoring tools should be in place to track resource usage and detect anomalies that could lead to cost overruns.
6. **Computation Power:** Utilizing specialized hardware, such as GPUs, is essential for accelerating the YOLOv7 model's inference. This accelerates detection speed and accuracy in critical situations.
7. **User Experience:** The application's user interface must be intuitive for military personnel, including those with limited technical expertise. Quick response times and clear feedback enhance usability.

**CHAPTER 7**

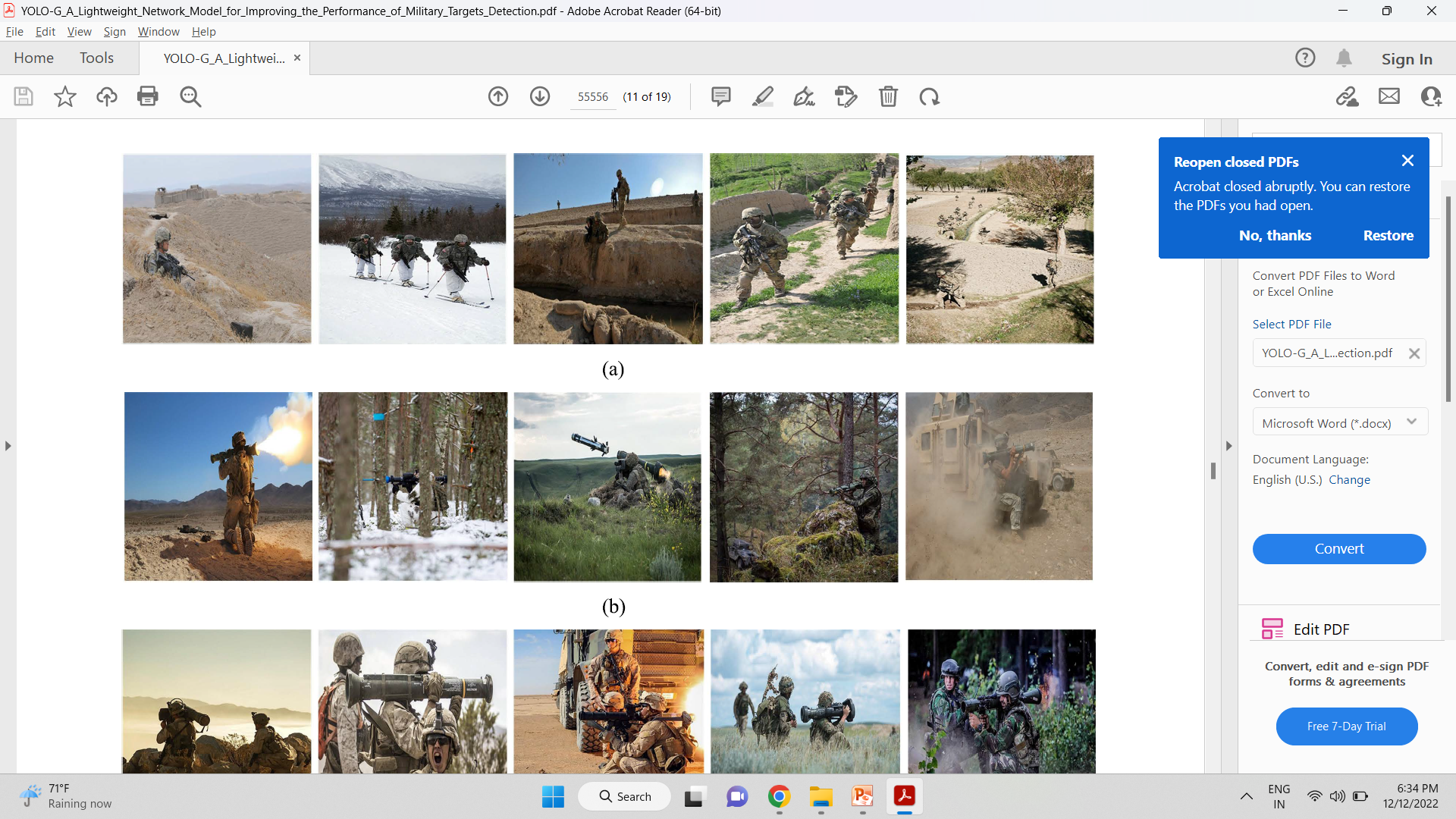
**SYSTEM IMPLEMENTATION**

### **Systems implementation** is a set of procedures performed to complete the design (as necessary) contained in the approved systems design document and to test, install, and begin to use the new or revised Information System. Implementation allows the users to take over its operation for use and evaluation. It involves training the users to handle the system and plan for a smooth conversion. The purpose of the implementation process is to design and create (or fabricate) a system element conforming to that element’s design properties and/or requirements. The element is constructed employing appropriate technologies and industry practices

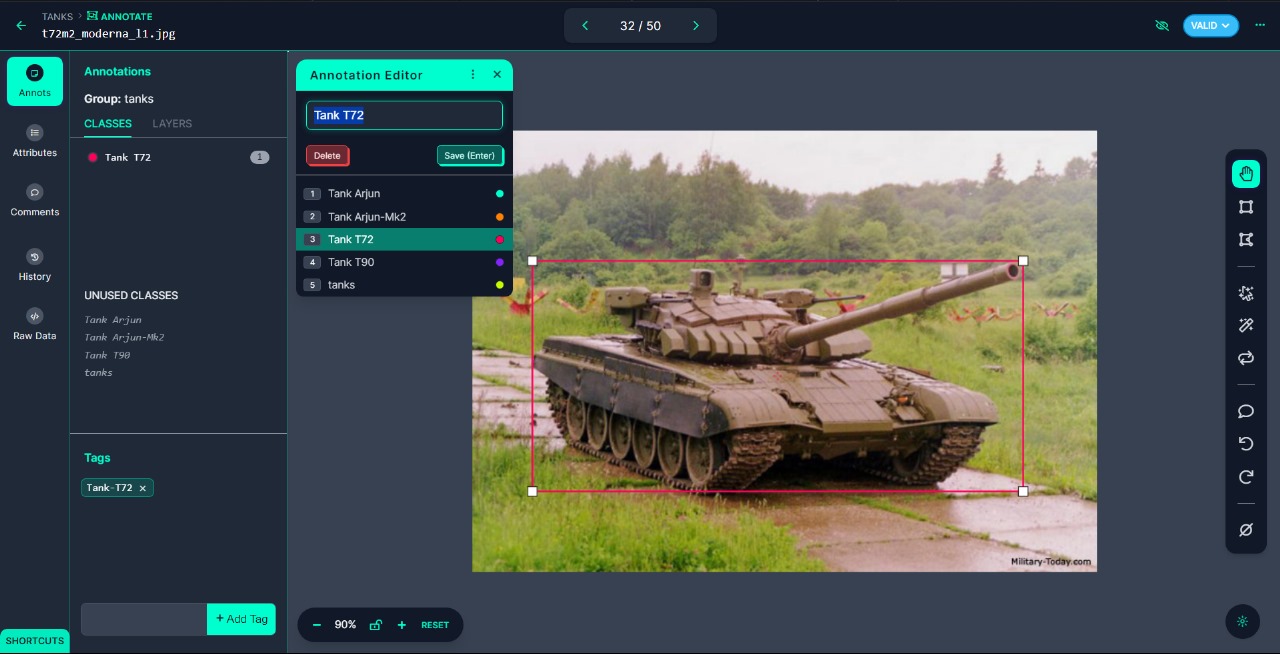
To begin, we will acquire a diverse dataset of military target and equipment images from Roboflow. This dataset will encompass a wide range of scenarios and will be meticulously annotated with bounding boxes to create ground truth labels. Additionally, the dataset will undergo preprocessing, including image resizing and normalization, to prepare it for training.

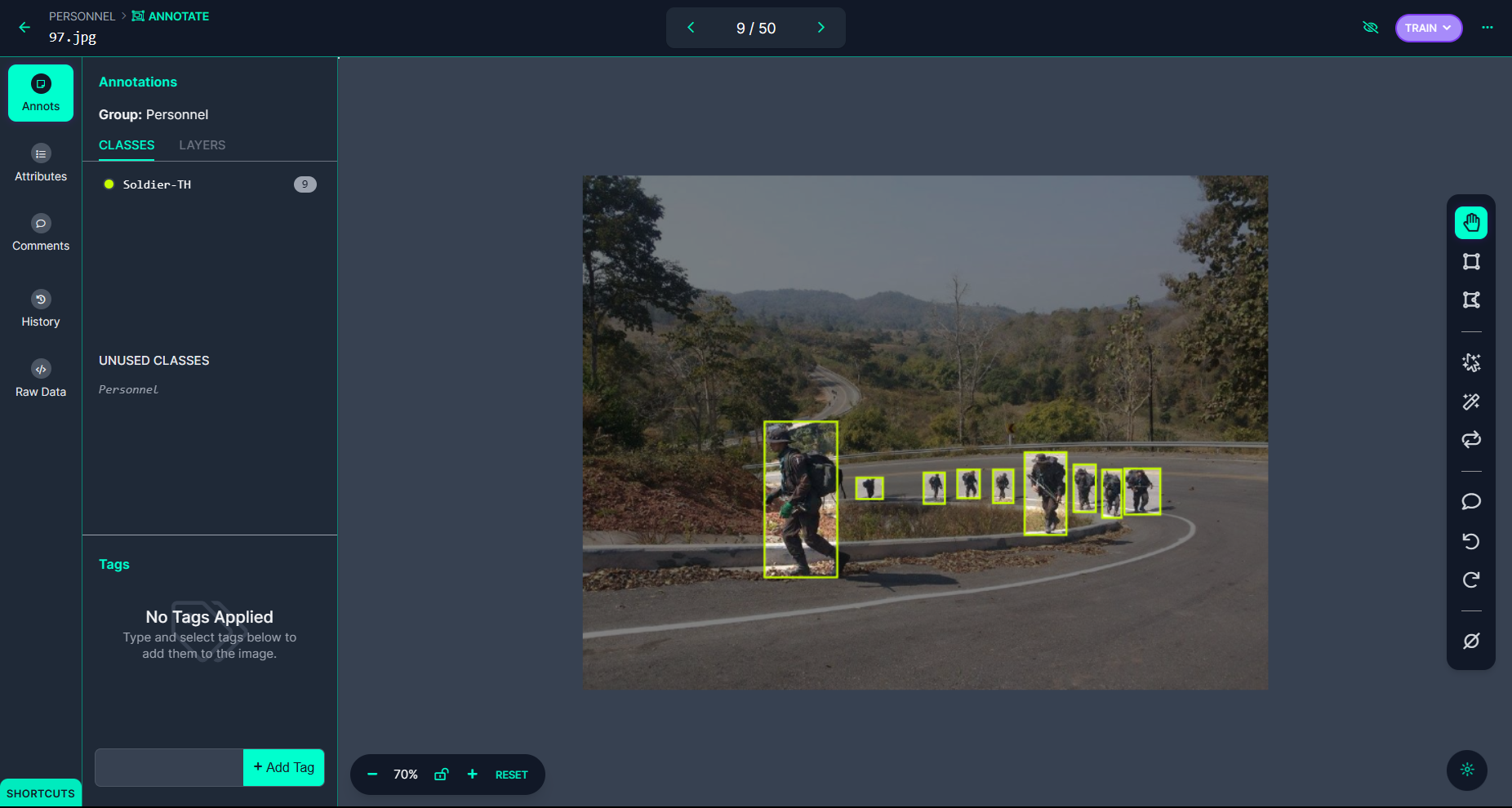
* 1. **Generated Standard Dataset using Roboflow :**

In this project, we used customized dataset to train the model. Constructed self-build dataset.

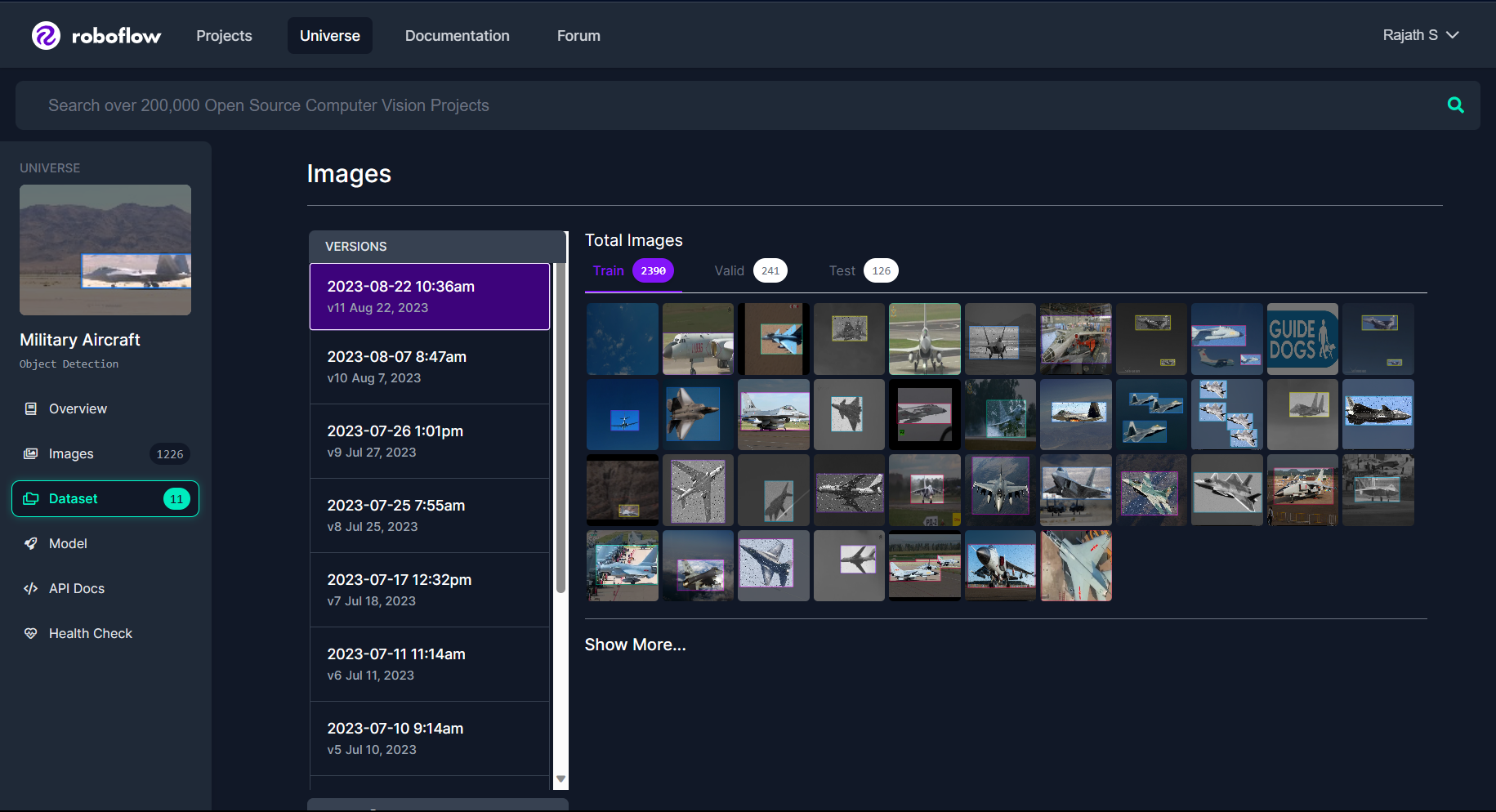


**Figure 7.1 Military Dataset**



7.2 Creating Dataset

7.3 Annotated classes in an Image



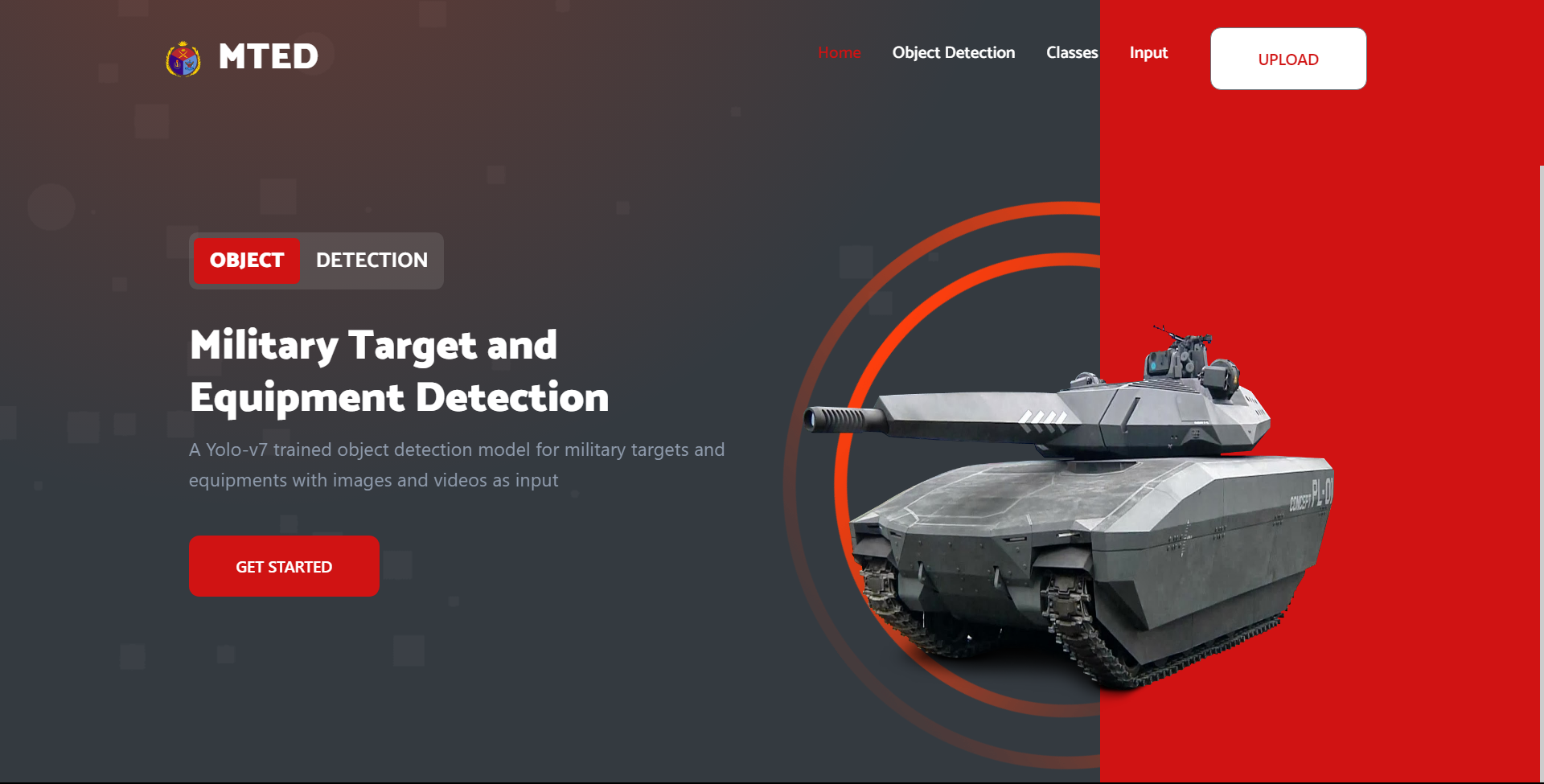
7.4 Generated Standard Dataset

The YOLOv7 architecture will serve as the foundational model for military target detection. The dataset will be split into training, validation, and test sets. Subsequently, we will train the YOLOv7 model using the training set, with a primary focus on optimizing precision and minimizing false positives. This training process may involve techniques such as transfer learning, data augmentation, and fine-tuning to enhance the model's performance.

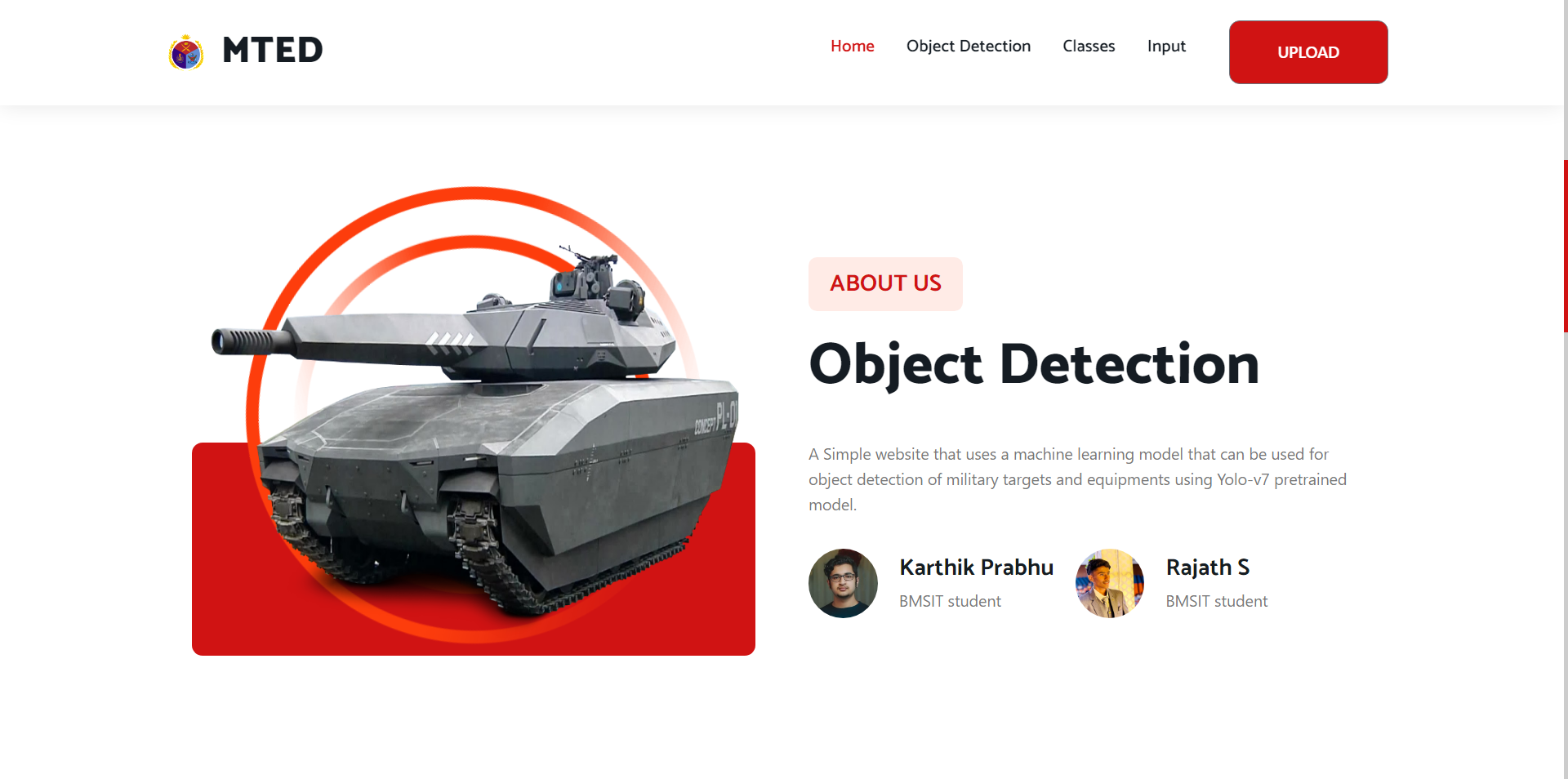
Following model training, we will thoroughly evaluate its performance. This evaluation will include assessing accuracy, precision, recall, and the F1 score using the validation and test datasets. Any shortcomings in detection performance will be meticulously analyzed, and improvements will be implemented as necessary. The development of the web application will be a pivotal aspect of this project. Using Flask, we will create a user-friendly interface that allows users to upload images for real-time military target detection. Integration with the trained YOLOv7 model will provide the core functionality of real-time detection within the application.

The user interface is designed with intuitiveness and responsiveness in mind to cater to the needs of military and security personnel. To adapt to the dynamic nature of military environments, mechanisms will be incorporated to account for factors like varying lighting conditions and terrains. These adaptations are essential to maintain the system's effectiveness in diverse operational scenarios. Finally, the deployment phase involves implementing the system in the intended operational environment, ensuring compatibility with various hardware configurations by running in the relevant environment of python. An ongoing maintenance plan will be established to address updates, enhancements, and evolving requirements in military target detection.

**7.2 Web application**



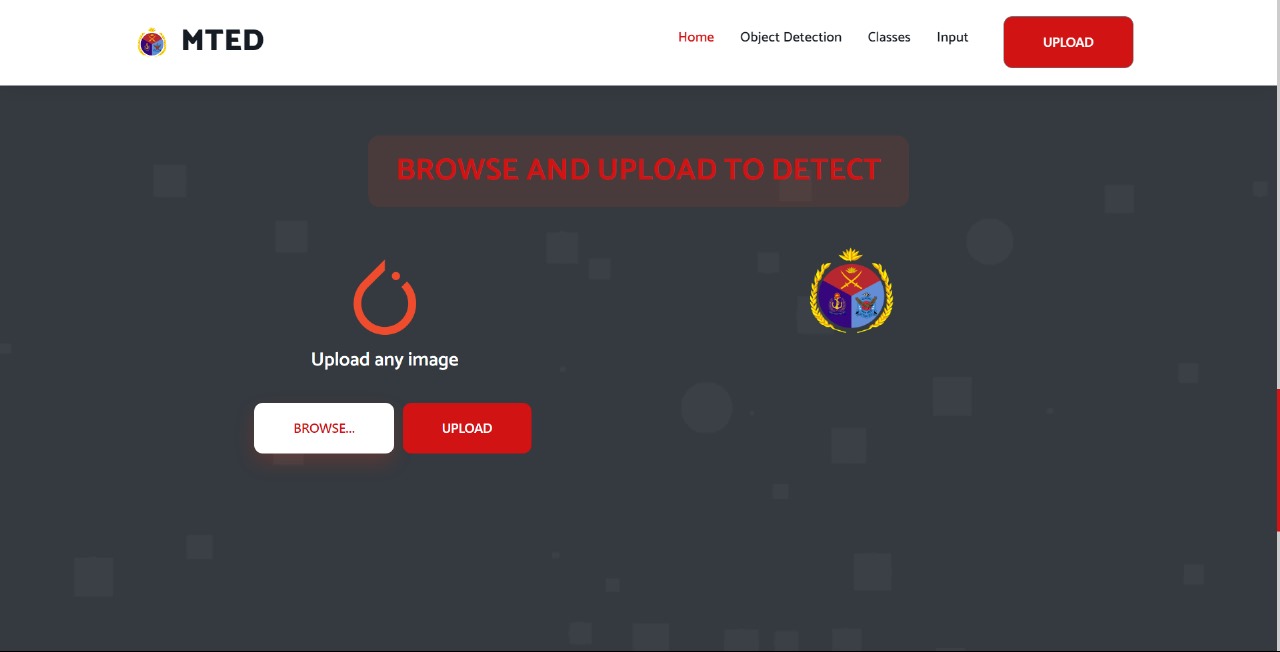
6.1 Landing Page of the Flask Application



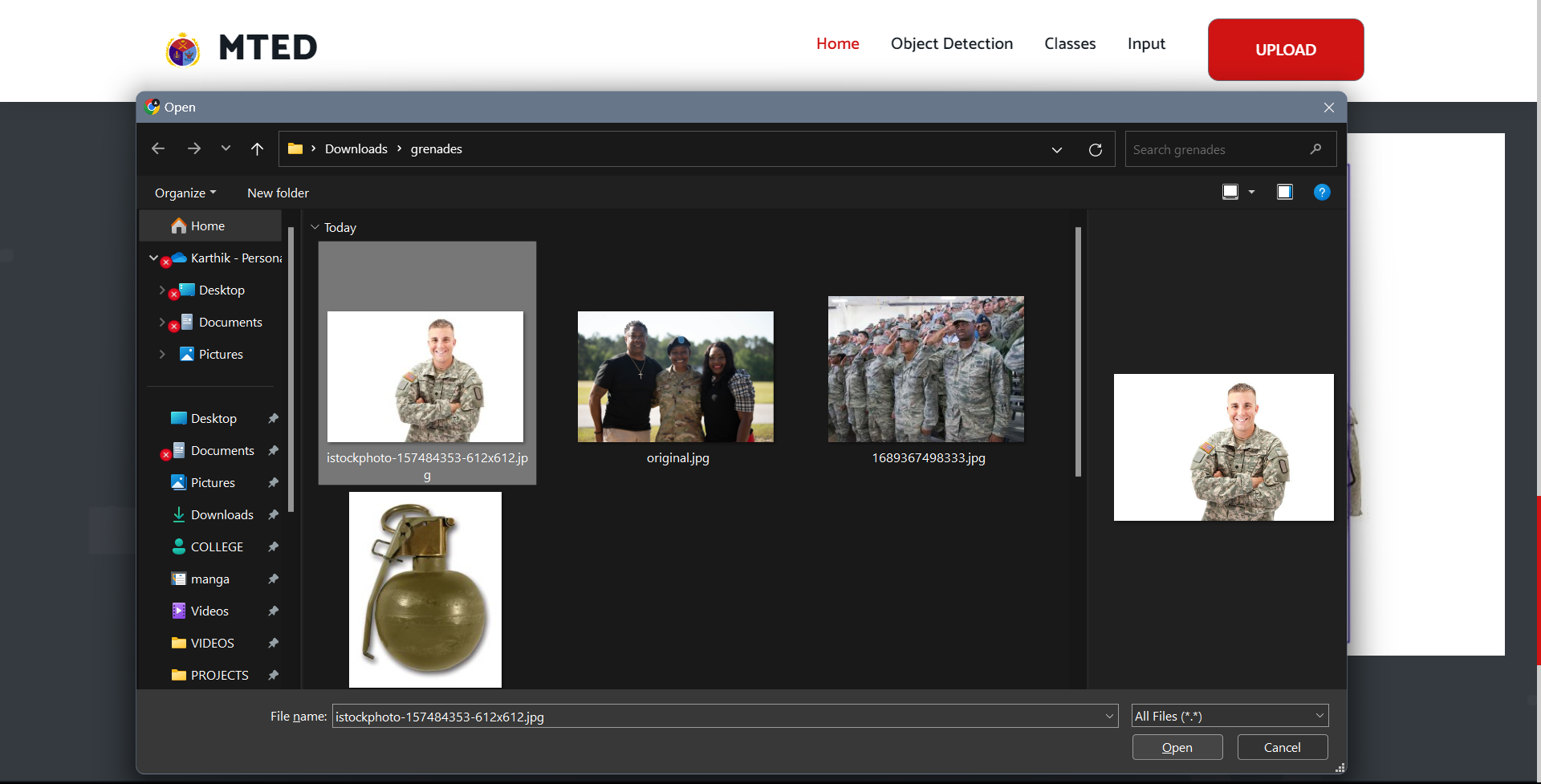
6.2 Object Detection Description for the Deep Learning Project



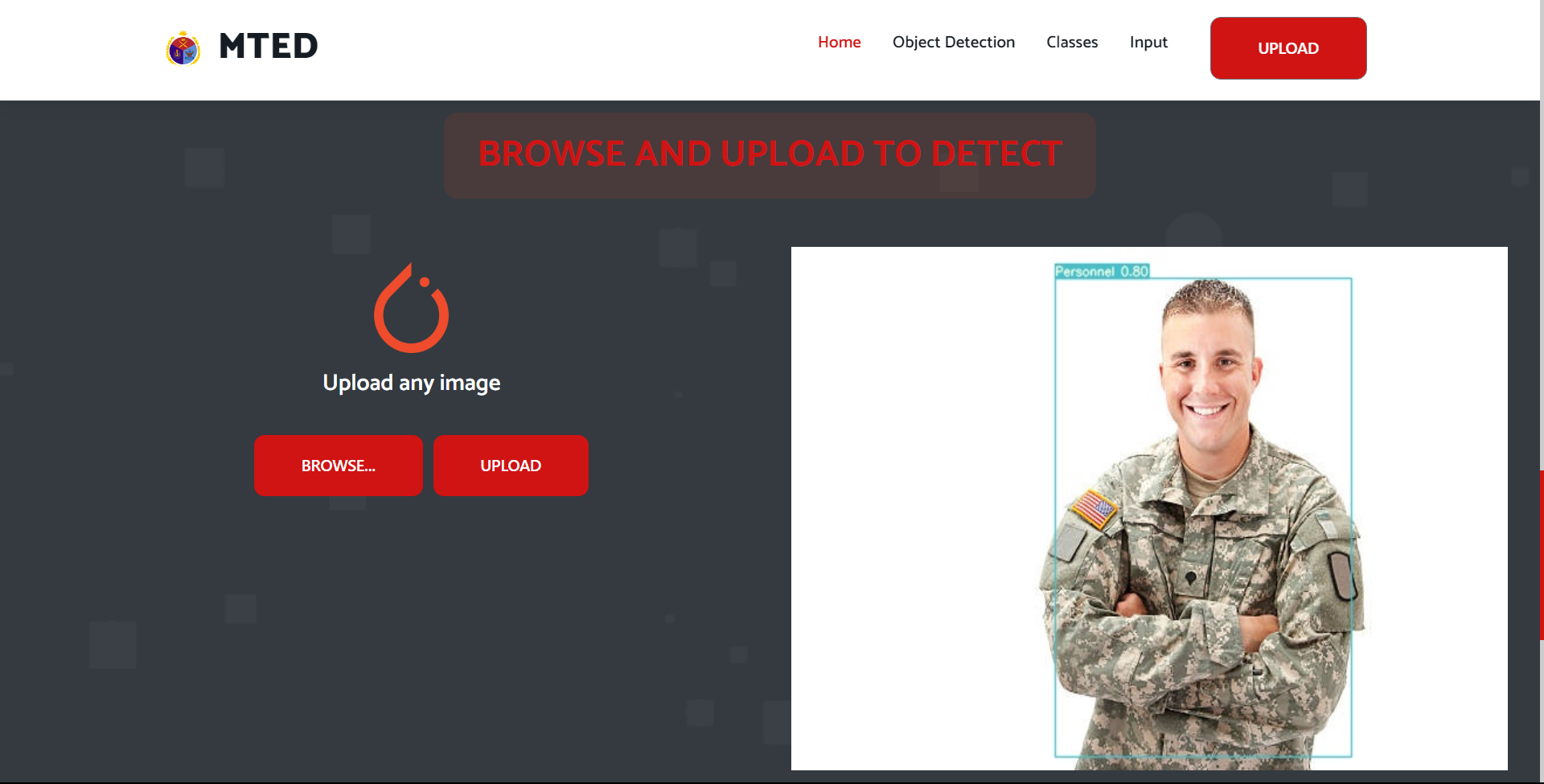
6.3 Classes Annotated for Object Detection in the flask Application



6.4 Upload Section to upload input images



6.5 Upload Pop-Up Window for the Input Image



6.6 Detected Image with Bounding box, class and confidence

**CHAPTER 8**

**SYSTEM TESTING**

Testing is extremely important for quality assurance and ensuring the products reliability. The success of testing for programmer flaws in largely determined by the experience. Testing might be a crucial component in ensuring the proposed systems quality and efficiency in achieving its goal. Testing is carried out at various phases during the system design and implementation process with the goal of creating a system that is visible, adaptable and secure.

Testing is an important element of the software development process. The testing procedure verifies whether the generated product meets the requirements for which it was intended.

**8.1 Test objectives**

• Testing may be a defined as a process of running a programme with the goal of detecting a flaw.

• An honest case is one in which there is a good chance of discovering a mistake that hasn’t been detected yet.

• A successful test is one that uncovers previously unknown flaw. If testing is done correctly, problems in the programme will be discovered. Testing cannot reveal whether or not flaws are present. It can only reveal the presence of software flaws.

**8.2 Testing principles**

A programmer must first grasp the fundamental idea that governs software testing before applying the methodologies to create successful test cases. All testing must be able to be tracked back to the customer’s specification.

**8.3 Testing design**

Any engineering product is frequently put to the test in one of two ways:

### 8.3.1 White Box Testing

### Glass container checking out is every other call for this kind of checking out. By understanding the necessary characteristic that the product has been supposed to do, checking out is regularly accomplished that proves every characteristic is absolutely operational at the same time as additionally checking for faults in every characteristic. The take a look at case layout technique that leverages the manage shape of the procedural layout to create take a look at instances is used on this take a look at case

### 8.3.2 Black Box Testing

### Tests are regularly finished on this checking out via way of means of understanding the indoors operation of a product to make certain that each one gears mesh, that the indoors operation operates reliably in step with specification, and that each one inner additives had been nicely exercised. It is in most cases worried with the software’s practical needs.

### 8.4 Testing Techniques

### A software testing template should be established as a set of stages in which particular test suit design techniques are defined for the software engineering process.

### The following characteristics should be included in every software testing strategy:

### ¬ Testing begins with the modules and extends to the mixing of the full computer based system.

### ¬ At different periods in time, different testing approaches are applicable.

### ¬ Testing is carried out by the software's developer and an independent test group. A software developer can use a software testing strategy as a route map. Testing might be a collection of actions that are prepared ahead of time and carried out in a methodical manner. As a result a software testing template should be established as a set of stages in which particular test suit design techniques are defined for the software engineering process. The following characteristics should be included in every software testing strategy: Testing begins at the module level and progresses to entire computer based system are mixing.

### ¬ At different periods in time different testing approaches are applicable.

### ¬ Testing is carried out by the software's developer and a separate test group.

### 7.5 Levels of Testing

### Testing is frequently omitted at various stages of the SDLC. They are as follows:

### 7.5.1 Unit Testing

### Unit testing checks the tiny piece of software that makes up the module. The white box orientation of the unit test is maintained throughout. Different modules are tested alongside the requirements created throughout the module design process. The aim of unit testing is to inspect the inner logic of the modules, and it is used to verify the code created during development phase. It is usually done by the module’s developer. The coding phase is sometimes referred to as coding and unit testing because of its tight association with coding. Unit tests for many modules are frequently run in simultaneously.

### 7.5.2 Integration Testing

### Integration testing is the second level of quality assurance. This type of testing integrates different components in program like modules also to check the interface problems. Many tested modules are combined into subsystems and tested as a result of this. The purpose of this test is to see if all of the modules are properly integrated. Integration testing may be divided into three categories:

### • Top-Down Integration:

### Top-Down integration is a method of gradually constructing a Programme structures. Modules are connected by working their way down the control Hierarchy, starting with the module having the most control. Bottom-Up Integration:

### Construction and testing using autonomous modules begin with Bottom-up integration, as the name suggests.

### • Regression Testing:

### It is a subset of previously executed tests to ensure that Modifications have not propagated unexpected side effects during this competition of an Integration test strategy.

### 7.5.3 Functional Testing; The business and technical requirements, system documentation, and user guides all specify that functional tests must be conducted to ensure that the functions being tested are available. The following items are the focus of functional testing:

### 7.5.4 Validation Testing: Validation may be characterized in a lot of ways; however one easy definition is that validation is a hit whilst software program plays in a manner that clients may fairly expect. The affordable expectation is said with inside the software program requirement specification that is a record that lists all the software program’s user-seen attributes. Validation standards are a segment of the specification. The statistics on this component serves as the premise for the validation trying out strategy.

### 7.5.5 Alpha Testing: Software developer can’t know how a customer will utilize a programme ahead of time. Instructions to be utilized could be misconstrued, a peculiar combination of knowledge could be employed on a regular basis, and a result that was clear to the tester could be unclear to a field user. It's impractical to conduct a formal acceptance test with all users if the programmed is designed as a product that will be used by many people. Most software developers utilize alpha and beta testing to detect bugs that only the most experienced users seem to be aware of. At the developer's premises, a customer does the trial.

**7.6 Test Cases**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Test case Objective** | **Prereq uisites** | **Steps** | **Input Data** | **Expected Output** | **Actual Output** | **Status** |
| 1 | Input the image | Read image | Upload the Image Identify the object or not | Military image | Object Detected |  | **Passed** |
| 2 | Input the image | Read image | Upload the Image the object or not | Military image | Object Detected |  | **Passed** |
| 3 | Input the Video | Read Video | Upload the Video to detect the object from clip | Military Video  Recognizing Firearms from Images and Videos in Real-Time with Deep Learning  and Computer Vision | by Tony Wang | Medium | Object Detected | Recognizing Firearms from Images and Videos in Real-Time with Deep Learning  and Computer Vision | by Tony Wang | Medium | **Passed** |
| 4 | Input Image | Read image | Upload the Image the object or not | Military image | Object Detected |  | **Failed** |

**CONCLUSION**

In summary, this research project has yielded a robust computer vision system for military target and equipment detection. Leveraging the YOLOv7 framework, pytorch, OpenCV, and Flask, we have created a versatile and user-friendly application. Notably, the ability to adjust the confidence threshold enhances precision customization, while support for real-time webcam and RTSP stream processing extends its utility.

This project bridges the gap between theory and practicality, equipping military and security personnel with a powerful tool to enhance operational efficiency and situational awareness. Its adaptability, precision, and security measures position it as a valuable asset in the defense and security domain, addressing critical needs in target identification and tactical decision-making.

**FUTURE ENHANCEMENT**

1. **Dynamic Confidence Threshold Adjustment:** Allowing users to adjust the confidence threshold for target detection provides them with enhanced control over the balance between precision and recall, enabling finer adjustments to meet specific operational needs. This flexibility accommodates a wider range of requirements with increased precision and adaptability.
2. **Integration of Webcam and RTSP Stream Support:** Integrating webcam and RTSP feed stream capabilities into the system would expand its functionality, enabling the real-time detection of targets within live video streams. This addition enhances the system's operational readiness, making it even more versatile and capable of performing target detection tasks in real-world scenarios.
3. **Class Identification and Confidence Reporting:** Enhancing situational awareness can be achieved by not only presenting identified objects but also providing their respective categories and linked confidence scores. This approach ensures users have a holistic understanding of detected targets and equipment, offering comprehensive information for improved decision-making and overall awareness in the given context.
4. **Streamlit Integration for Enhanced User Interface and Settings:** Integrating the Streamlit framework holds promise for creating an interactive and highly adaptable user interface. By harnessing Streamlit's capabilities, users gain the power to fine-tune detection settings, thus fostering a more personalized and tailored experience to meet their specific needs.

**REFERENCES**

[1] M. Everingham, L. Van Gool, An Incremental Improvement 2018

[2] H. Peng, Y. Zhang, S. Yang, And B. Song, ``Battlefield Image Situational Awareness Application Based On Deep Learning,'' *IEEE Intell. Syst.*, Vol. 35, No. 1, Pp. 36\_43, Jan. 2020.

[3] Zhongzhen Sun , Muchen Dai, Xiangguang Leng *, Member, Ieee*, Yu Lei, Boli Xiong, An Anchor-free Detection Method For Ship Targets In High-resolution Sar Images, Ieee Journal Of Selected Topics In Applied Earth Observations And Remote Sensing, Vol. 14, 2021

[4] Xiaoru Song, Song Gao \*, Chaobo Chen “An Improved One-stage Pedestrian Detection Method Based On Multi-scale Attention Feature Extraction” Ieee Transactions On Journal 2021.

[5] LINGREN KONG 1, JIANZHONG WANG 1, AND PENG ZHAO 2, “YOLO-G: A Lightweight Network Model For Improving The Performance Of Military Targets Detection” IEEE ACCESS (2022)

[6] A. K. Jain, N. K. Ratha, And S. Lakshmanan, ``Object Detection Using Gaborlters,'' *Pattern Recognit.*, Vol. 30, No. 2, Pp. 295\_309, 1997.

[7] S.-G. Sun, ``Automatic target detection using binary template matching,'' *Opt. Eng.*, vol. 44, no. 3, Mar. 2005, Art. no. 036401.

[8] V.-E. Neagoe, S.-V. Carata, and A.-D. Ciotec, ``An advanced neural network-based approach for military ground vehicle recognition in SAR aerial imagery,'' *Int. Sci. Committee*, vol. 18, p. 41, Jan. 2016.

[9] W. Budiharto, V. Andreas, J. S. Suroso, A. A. S. Gunawan, and E. Irwansyah, ``Development of tank-based military robot and object tracker,'' in *Proc. 4th Asia\_Paci\_c Conf. Intell. Robot Syst. (ACIRS)*, Jul. 2019, pp. 221\_224.

[10] A. Krizhevsky, I. Sutskever, and G. E. Hinton, ``ImageNet classi\_cation with deep convolutional neural networks,'' in *Proc. 25th Int. Conf. Neural*

*Inf. Process. Syst.*, 2013, pp. 1097\_1105.

**APPENDIX I**

**ABBREVATIONS**

**FPN:**  feature pyramid network

**YOLOv3:** You Look Only Once, Version 3

**ATD:** Automatic Target Detection

**CCD:**  Charge Coupled Device

**SAR:**  Synthetic Aperture Radar

**RPN:** Region Proposal Network

**RCNN:** regions with convolutional neural networks

**SRS:** System Requirements Specification

**IDLE:** Integrated Development and Learning Environment

**UML**: Unified Modeling Language